Utilizando una Estructura de Árbol para un ChatBot de Recomendaciones

Sistemas Inteligentes

Sánchez Patiño Natalia
Rosas Otero Mario
Velázquez Vilchiz Mario

Profesor: Dr. David Tinoco Varela

Licenciatura en Tecnología e ITSE
Facultad de Estudios Superiores Cuautitlán
Universidad Nacional Autónoma de México

05 - 12 - 2020

-Proyecto para Sistemas Inteligentes: Aplicación a las Estructuras de Árboles mediante un ChatBot.-

Resumen

Chatbot de recomendaciones basado en una estructura de árbol. En este proyecto se realiza la construcción de un sistema capaz de dar recomendaciones en áreas académicas y de entretenimiento mediante una comunicación en lenguaje español de forma natural teniendo la posibilidad de interactuar mediante medios visuales con entrada de información escrita, así como interactuar de forma audiovisual con el usuario. Realizando así una implementación que fluye a tres variantes del sistema de forma directa. El correcto funcionamiento de las versiones construidas requiere de elementos tanto de hardware como de software, e involucra mantener distintos archivos, de ejecución de código, y bases de datos. Dichos archivos en conjunto permiten tener la información necesaria para brindar una experiencia de recomendación, con el objetivo de que el usuario al utilizar el sistema descubra trabajos particulares en áreas de interés para el, como lo pueden ser artículos científicos, música, vídeos, películas, definiciones de conceptos, entre otros. Con esto realizamos la propuesta de una idea a pequeña escala pero que pueda ser transformada a la solución de una situación cotidiana, donde se requiera mejorar la experiencia de comunicación humano-maquina. De esta forma es como decidió implementar este proyecto utilizando conceptos vistos en nuestra clase de sistemas inteligentes, especialmente tomamos una estructura del árbol para controlar el funcionamiento general del ChatBot, buscando bases de datos que fueran relacionadas a conceptos y actividades a fines a la mayoría de la población. Teniendo como uno de los objetivos prioritarios el que fuese de fácil accesibilidad con la única condicionante de tener conexión de internet durante su uso en linea, o con la posibilidad de usarlo sin conexión, ejecutando el sistema de forma local, con la implicación de que se debe realizar una configuración inicial mas laboriosa, en el dispositivo donde se ejecuta. Dentro de las posibles extensiones del proyecto se plantea el funcionamiento del sistema dentro de una tarjeta de desarrollo como Raspberry Pi, a fin de dar portabilidad al sistema e integrarlo a otros dispositivos, con posibilidad de extensión de características, siendo posible integrarlo a dispositivos como controladores de casas inteligentes. Es posible obtener el código total del proyecto en: https://github.com/NM-Labs/ChatBot

Índice

1 Introducción **1**

1.1 Motivación 1

1.2 Objetivos 2

1.3 Solución Propuesta 2

2 Marco teórico **4**

2.1 Arboles de decisión 4

2.2 ChatBots 4

3 Elementos Utilizados y Proceso de Construcción **7**

3.1 Hardware 7

3.2 Software 8

3.3 Funcionamiento del Sistema 8

3.3.1 Parte A: ChatBot Escrito en Colab 11

3.3.2 Parte B: ChatBot Hablado en ambiente local 12

3.3.3 Parte C: ChatBot Hablado en Raspberry 12

4 Resultados obtenidos **14**

4.1 Practicidad de los Ambientess Propuestas 14

4.1.1 Google Colaboratory y GitHub 14

4.1.2 Ambiente local: Computadora 15

4.1.3 Ambiente local: Tarjeta de Desarrollo 15

4.2 Interacción con el ChatBot 16

4.2.1 Seguimiento y control de la conversación 16

4.2.2 Entrada y salida de información 16

5 Conlusiones **18**
5.1 Posibles extensiones del proyecto 19
5.2 Comentarios Finales 20

Appendix A Codigo de MMN ChatBot con Entrada Escrita **21**

Appendix B Codigo de MMN ChatBot con Entrada Hablada **43**

Referencias **67**

1

Introducción

El proyecto desarrollado plantea la construcción de un ChatBot capaz de dar recomendaciones, dichas recomendaciones pueden implicar descubrimientos en diferentes actividades de acuerdo a las preferencias o a los intereses del momento que tiene el usuario, de igual forma el ChatBot, nombrado como MMN ChatBot tiene la capacidad de brindar información actualizada acerca de COVID en México, así como brindar un mapa interactivo acerca de la evolución de COVID en el mundo.

1.1 | Motivación

Una de la tareas mas favorecedoras, y que mas se buscan para establecer contacto y guiar a los usuarios en alto nivel a través de un sistema computacional es la mejora de interacción humano-computadora, el desarrollo y mejora de sistemas computacionales capaces de comportarse como un humano lo considera de forma *familiar*, puede ser ventajoso en distintos ámbitos como los negocios o la educación. El uso de Chatbots, bajo el concepto de ser programas capaces de comunicarse con un lenguaje humano, en uno o varios idiomas y que intenta que la experiencia del usuario a través de un sistema computacional sea mas amigable, productiva, y se genere un *vinculo de confianza*, mayor al que se tiene al interactuar con una computadora, se han popularizado, y con esto, han aparecido desarrollos e innovaciones al rededor de los mismos, tal que se ha aumentado la capacidad de sus acciones si es que tiene permisos de modificación, creación o supresión de archivos y acciones en determinado sistema. La mayoría de los Chatbots, son utilizados como asistentes personales, para automatizar tareas, o que las indicaciones de acción se den de forma mas natural, ya sea en entrada escrita, entrada por voz, entre otros. Algunos otros Chatbots tienen la función de ser guías a través de un sistema especifico. En ocasiones es difícil encontrar Chatbots con otras finalidades, es por eso

que se planteo hacer un Chatbot cuya finalidad es dar recomendaciones para distintos ámbitos dentro del entretenimiento, y el ámbito académico, así como para brindar información actualizada de la pandemia generada por el COVID-19.

1.2 | Objetivos

- **Objetivo general:**
  Construir de forma práctica un sistema ChatBot de recomendaciones completo utilizando un concepto visto en la clase, que son los arboles, se requiere que el sistema sea capaz de recibir entrada escrita o por voz, para incentivar nuevos descubrimientos en áreas de interés del usuario, brindando en muchas ocasiones enlaces a direcciones web para complementar la información o la experiencia del usuario. De igual forma, en algunos casos se brindan herramientas interactivas, dentro del sistema.

- **Objetivos Particulares:**

  - Realizar una revisión teórica acerca del concepto de estructuras de arboles.
  - Buscar la aplicabilidad del concepto de árbol como estructura de organización de datos, como estructura general de funcionamiento y control para dirigir una conversación.
  - Comprender de manera teórica y práctica, mediante un lenguaje de programación, el funcionamiento del sistema propuesto.
  - Complementar y desarrollar habilidades de programación en el lenguaje Python como herramienta para la creación y ejecución del sistema.
  - Verificar que el sistema cuente con un funcionamiento práctico, así como identificar e implementar el sistema con herramientas del fácil acceso, así como abiertas a todo publico.
  - Comprobar de manera aplicada, conceptos vistos en clase y conjuntandolo con conceptos encontrados durante la búsqueda de información relacionada a temas de la clase.

1.3 | Solución Propuesta

Para la implementación de un sistema capaz de llevar una interacción natural, y establecer un buen esquema de comunicación, siendo capaz de dirigir una conversación

donde se brindan recomendaciones, se propone el planteamiento del sistema en el que el funcionamiento general de dirección de conversación esta dado por una estructura de árbol. Teniendo tres esquemas diferentes, que representan escenarios en los que el sistema puede funcionar de forma correcta. El primero es utilizando herramientas de almacenamiento en linea como GitHub, y herramientas que ofrecen una interfaz gráfica de ejecución de código escrito en Python de igual forma en linea, donde el Chatbot presenta un tipo de entrada. El segundo escenario de funcionamiento es en ejecución local en una computadora mediante el software de Anaconda, el cual brinda la capacidad de ejecución de código en Python en una interfaz gráfica amigable, y nos permite crear un ambiente de ejecución independiente con las herramientas necesarias para el correcto funcionamiento del sistema, y teniendo la capacidad de acceder a los altavoces y al micrófono del dispositivo teniendo así la capacidad de interactuar mediante voz con el Chatbot. El ultimo escenario siendo similar al anterior teniendo la capacidad de ejecutar el sistema de forma local desde una tarjeta de desarrollo, en este caso se plantea el uso en una Raspberry Pi, la cual cuenta con los puertos necesarios para tener entrada y salida de audio, y salida visual, con esto poder ejecutar el sistema con reconocimiento de voz. pero todo mediante la ejecución desde una tarjeta de desarrollo.

2

Marco teórico

2.1 | Arboles de decisión

Un árbol de decisiones es un algoritmo que se puede utilizar para problemas de regresión y clasificación; sin embargo, se utiliza principalmente para problemas de clasificación. Un árbol de decisión sigue un conjunto de condiciones if-else para visualizar los datos y clasificarlos según las condiciones.
Un árbol de decisión es un diagrama de flujo como una estructura de árbol, donde cada nodo interno denota una prueba en un atributo, cada rama representa un resultado de la prueba y cada nodo hoja (nodo terminal) tiene una etiqueta de clase. Es decir mediante la cuantificación o categorización de variables que intervienen en el problema planteado, el árbol nos ayuda a seguir un camino de evaluaciones a los datos de entrada, que al final nos llevan a la conclusión del análisis de la información de entrada teniendo una salida que dará respuesta o conclusión del análisis realizado a dicha información que pasa a través de la estructura del árbol. En la figura 2.1, es posible ver un diagrama que representa e indica los componentes y la forma de una estructura de árbol para datos.

2.2 | ChatBots

Los ChatBots muy populares hoy en día y está ganando velocidad en su uso como herramientas de comunicación informática, específicamente para establecer comunicación humano-máquina. Algunos de estos sistemas intentan responder de forma *inteligentemente*, es decir intentando semejar un comportamiento humano. Un Chatbot puede ser implementado mediante la comparación de patrones, en la que se reconoce el or-

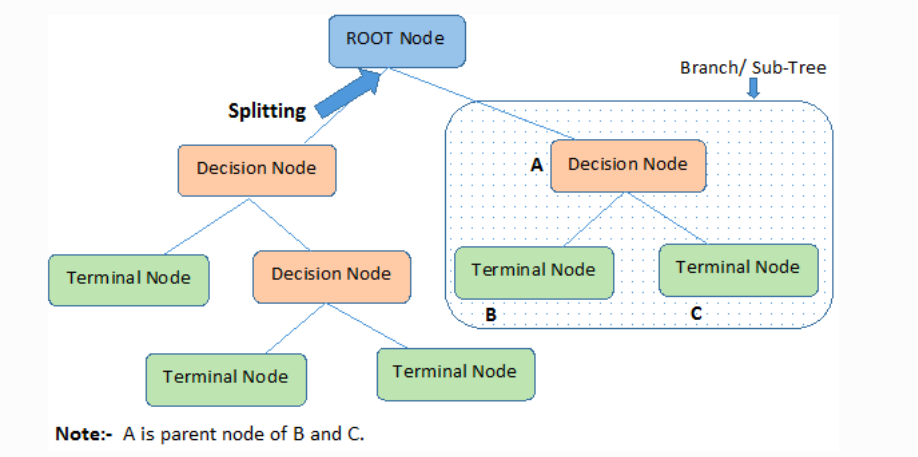


Figura 2.1: Representación de un árbol de decisión

den de la oración y un patrón de respuesta guardado aclimata a las variables exclusivas de la oración, intentamos identificar palabras claves en la entrada recibida del usuario para tomar acción respecto a dicha entrada. Hasta este momento, aun no pueden mantener conversaciones generales de forma totalmente adecuada, ni responder a preguntas complejas, así como realizar actividades compuestas. La mayoría de los Chatbots están implementados para tareas especificas como guías y asistentes, con la que se realiza automatización de tareas, o para brindar información.

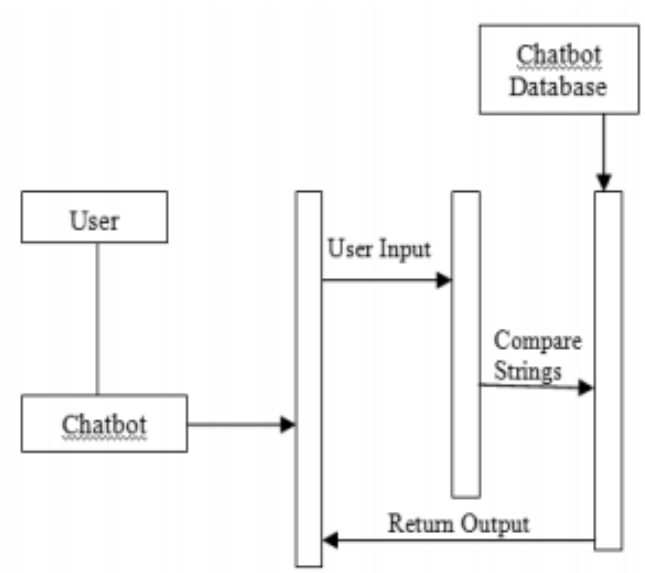


Figura 2.2: Diagrama de secuencia que representa el diseño de un Chatbot

La sesión de chat, es decir la conversación de un usuario con el sistema, es esencialmente una serie de preguntas y respuestas, donde se tiene un guión que intenta dirigir una conversación para poder obtener la información necesaria y que conducen a un diagnóstico. Entonces podría implementarse en un árbol de decisiones. En cada nodo del árbol de decisión, el sistema hace una pregunta típicamente de opción múltiple donde se intenta obtener información para conocer el interés del usuario, y según la respuesta, se ramifica al siguiente nodo, con esto se dirige el sistema a los nodos próximos a lo que el usuario busca para obtener un diagnostico y dar una respuesta .

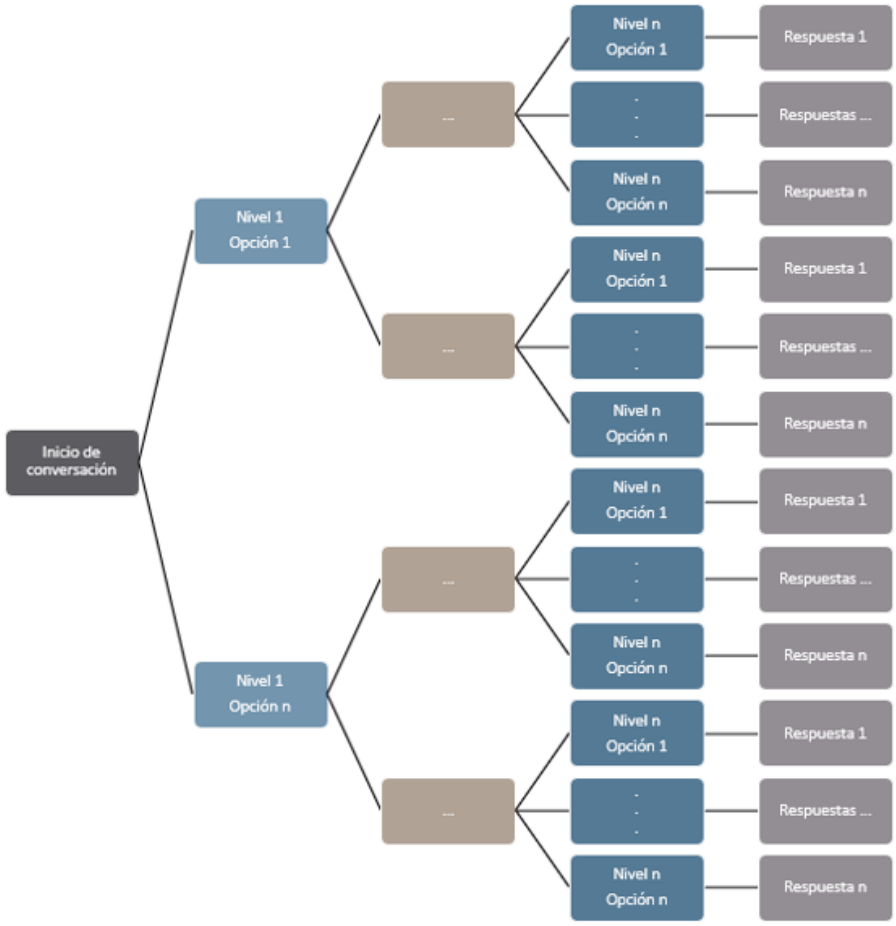


Figura 2.3: Representación de un árbol para modelar una conversación de recomedaciones.

3

Elementos Utilizados y Proceso de Construcción

Para la creación de nuestro proyecto se utilizó un conjunto de componentes tanto de software como de hardware, para la Parte A del proyecto que es un ChatBot con interacción puramente textual solo contempla elementos de software en línea, como lo es la plataforma ”Google Colaboratory (Colab)” y el gestor de repositorios ”GitHub”, que nos permiten realizar todo el proceso de funcionamiento del sistema en linea, y desde cualquier equipo, siempre y cuando el navegador este actualizado y permita el correcto funcionamiento de dichas plataformas en línea. Para la Parte B y Parte C parte del proyecto (ChatBot con entrada y salida de voz) se contemplan los siguientes elementos:

3.1 | Hardware

- Parte B
  - Computadora con micrófono y altavoz habilitados.
  - Audífonos con micrófono (opcional)
- Parte C
  - Placa de desarrollo RaspBerry.
  - Periféricos necesarios para el funcionamiento de la RaspBerry:
    - Mouse
    - Teclado
    - Monitor
  - Audífonos con micrófono.

3.2 | Software

- Parte B y Parte C
  - Anaconda - Jupyter
  - Un ambiente independiente para Python 3 en anaconda para la descarga de librerías necesarias.

3.3 | Funcionamiento del Sistema

La funcionalidad en ambos sistemas es sumamente similar, la característica distintiva es la manera de interactuar con el usuario. Las funciones con las que cuenta MMN ChatBot son dar recomendaciones acerca de:

- Recomendaciones de entretenimiento
  - Vídeos
  - Películas
  - Series
  - Música
  - Libros
  - Videojuegos
- Recomendaciones académicas
  - Artículos
  - Investigadores
  - Definiciones
- Información sobre covid

Todo esto siguiendo el esquema de la figura 3.1. Una vez que se activa el funcionamiento del ChatBot creando un objeto a partir de la importación del script, y solicitando una conversación con `c.chatear()` comienza la interacción. Primeramente MMN ChatBot saluda, se presenta y pregunta tu nombre. Con esto analiza las palabras en la respuesta, y realiza la comparación con la base de datos de nombres, si encuentra alguna coincidencia y logra detectarlo, partir de ese momento, en la parte de la impresión de la

conversación del usuario aparecerá su nombre.
Nuestro diagrama de flujo se puede ver en la figura 3.2

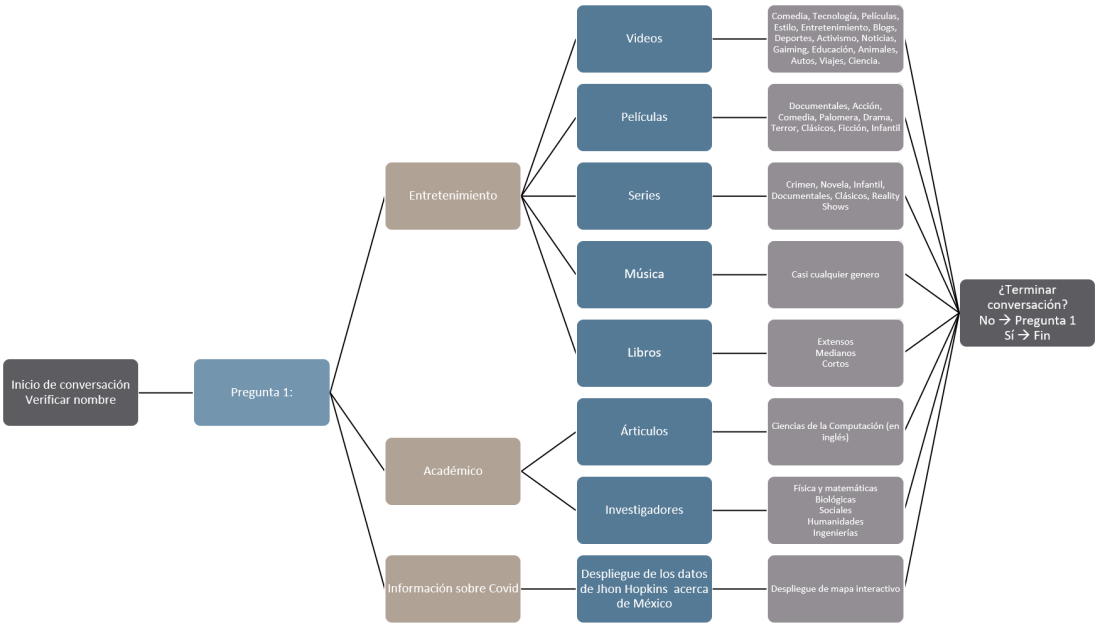


Figura 3.1: Árbol del Chatbot

Posterior al inicio de interacción, el ChatBot debido a su estructura guía al al usuario a través de ramificaciones realizando preguntas acerca de los intereses del momento del usuario y brindando opciones, que quedara a elección del usuario el definir el camino a seguir a partir de las opciones brindadas por el ChatBot, y el cual lo llevara a generar una recomendación dados los datos obtenidos durante la conversación. Dos de las tres categorías de recomendación general, que son *entretenimiento* y *académico* tienen a su vez, una ramificación hacia subcategorías, donde se dan opciones de actividades particulares relacionadas a las categorías generales. Una vez realizando una elección dentro de las opciones de subcategorías, se requiere que el usuario ingrese información acerca de sus preferencias específicas dentro de dicha actividad, para posteriormente buscar información en la base de datos correspondiente que coincidan con las características brindadas por el usuario como sus preferencias, una vez que se encuentran las coincidencias en la base de datos correspondiente, se realiza una elección aleatoria para hacer la recomendación. En algunos casos, donde las bases de datos así lo permiten, es posible brindar al usuario enlaces a otras paginas donde se puede acceder por completo

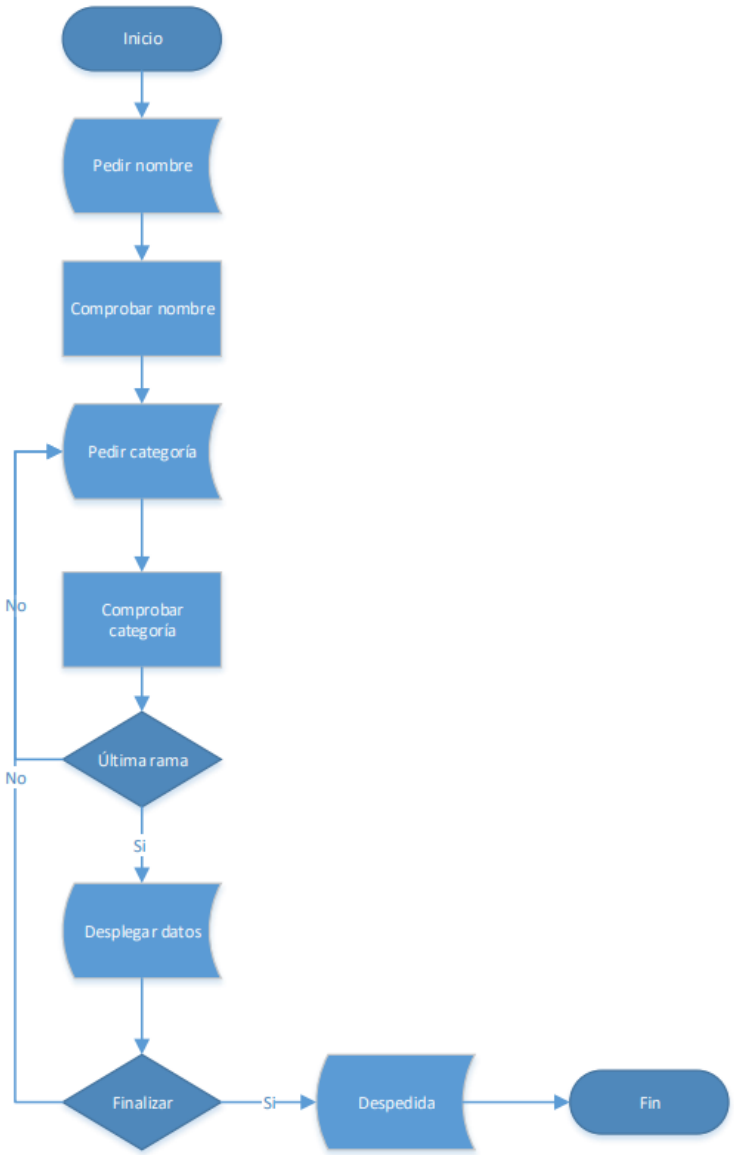


Figura 3.2: Diagrama de flujo del Chatbot

a la recomendación realizada o a una profundización de información sobre la misma. Una ves que se obtiene una recomendación, que en este caso representa un nodo terminal en el árbol, se pregunta al usuario si quiere seguir interactuando y obtener mas recomendaciones o desea finalizar la conversación. Con esto si el usuario desea seguir la conversación el ChatBot vuelve a brindar las opciones desde las categorías generales de recomendación, para que el usuario elija un camino nuevamente, de lo contrario si

el usuario elije terminar la conversación el ChatBot se despide y finaliza el proceso. En el nodo de categorías generales también se encuentra una sección donde se brinda información actualizada al día en que se ejecuta el código sobre el estado de Covid en México, y se muestra un mapa mundial, sobre el que se ejecuta una linea de tiempo de la evolución de Covid en el mundo hasta un día antes de la ejecución de código, dicho mapa es interactivo y si pone el puntero sobre algún país brinda información de Covid en dicho lugar, además es posible realizar desplazamientos sobre dicho mapa.

La construcción general del ChatBot se compone de las bases de datos, un script python donde se realiza la construcción general del funcionamiento del chatbot, y además se compone de un notebook python desde el que se ejecuta el funcionamiento del ChatBot, y nos brinda una interfaz de usuario amigable y minimalista con el cual la ejecución del Chatbot se puede integrar de forma estética y fácil de poner en acción.

3.3.1 | Parte A: ChatBot Escrito en Colab

Como se dijo anteriormente, la intención inicial de este proyecto fue que cualquiera pudiera interactuar con nuestro ChatBot en cualquier dispositivo, es por eso que se decidió crear una versión con ejecución en la plataforma de Colab que permite correr notebooks de Python en línea aprovechando los recursos de los servidores de Google. En este respecto, nos encontramos con el problema mayor, de que no es posible ejercer la acción de reconocimiento de voz desde esta plataforma dado que se ejecuta sobre maquinas virtuales, y el manejo del micrófono desde el dispositivo que se controla el notebook es limitado, por ello se decidió dejar una versión únicamente con entrada de texto escrito, y crear una versión extra para correr en ambiente local de los dispositivos, la cual es capaz de recibir la información y guiar la conversación de forma auditiva.

La primera parte contempla el funcionamiento del ChatBot a través de la escritura directa de las respuestas. Para interactuar con el se puede acceder a la página `https://colab.research.google.com/github/NM-Labs/ChatBot/blob/main/Chatbot.ipynb`. Los códigos tanto del script como del notebook de python para ejecución se pueden encontrar en el GitHub del proyecto, y además una versión del texto contenido en este mismo documento en el (Anexo A). Si se realiza la ejecución del sistema en Colab, es necesario correr la primera celda oculta, esto lo que hará será descargar las librerías y archivos necesarios, y a continuación correr la celda con el nombre ”MMN ChatBot”.

3.3.2 | Parte B: ChatBot Hablado en ambiente local

De manera similar a el ChatBot Escrito, el ChatBot Hablado trabaja a través de notebooks de Python consultando un script con extensión *.py*, pero en esta ocasión la interacción con el ChatBot se produce por medio de la voz, por lo que se necesita tener un micrófono y unos altavoces o audífonos, en la versión en línea esto no era posible ya que el programa no reconocía ningún micrófono al que pudiera acceder, lo que hacía imposible que el ChatBot pudiera escuchar las respuestas dadas. Para solucionar este problema se decidió utilizar la distribución de Anaconda local. Para esto es necesario de igual manera tener varias librerías instaladas, estas se detallan en la primera celda del archivo *ChatbotSR.ipynb* y pueden verse también en la sección del notebook en el (Anexo B). Esta celda no es necesario correrla siempre que se vaya a conversar con el ChatBot, sino una única vez cuando se usa por primera vez. Todos los archivos del sistema se encuentran disponibles en GitHub que se encuentran en la página `https://github.com/NM-Labs/ChatBot`.

En esta versión el ChatBot, además de captar las respuestas del usuario directamente desde reconocimiento de voz, es también capaz de utilizar voz sintética para responder al usuario, adicional a ello, se conserva la impresión en pantalla del registro de la conversación, y donde el usuario puede reforzar la comunicación de forma visual, además esto conserva de forma natural que los enlaces a otras páginas brindados se puedan acceder con solo un clic. Adicional a esto, se imprimen en pantalla claras y llamativas indicaciones de cuando el Chatbot esta hablando, cuando esta escuchando y cuando esta procesando la información, a fin de indicar de manera precisa el estado de la conversación en tiempo real, y dirigir los tiempos de habla y escucha entre ambos.

3.3.3 | Parte C: ChatBot Hablado en Raspberry

El funcionamiento es el mismo que en la Parte B que ya describimos, pero esta vez se utiliza una tarjeta de desarrollo Raspberry. Para poder usar los notebooks de Python dentro de la tarjeta tenemos que seguir estos pasos:

- Preparar el ambiente de Python
- Instalar JupyterHub
- Configurar como servicio del sistema
- Inicializar JupyterLab

- Instalar bibliotecas de Python
- Descargar o clonar los archivos necesarios del repositorio de GitHub.

Una vez hecho esto, puesto que la interacción con el ChatBot se produce por medio de la voz, será necesario conectar un micrófono y bocinas o audífonos. El funcionamiento se hará de manera similar a lo descrito anteriormente en la parte B. Añadido a esto, si se contase con un modulo LCD seria posible mostrar la conversación en pantalla. Demostrando con esto la viabilidad de implementación del sistema del ChatBot en diferentes tipos de dispositivos, y encontrar aplicación tangible del sistema propuesto.

4

Resultados obtenidos

4.1 | Practicidad de los Ambientess Propuestas

4.1.1 | Google Colaboratory y GitHub

El uso de google Colaboratory en conjunto con GitHub, es una gran opción para mantener en linea y al alcance de todos en la red nuestro Chatbot. Colaboratory nos brinda un entorno amigable de fácil acceso, y con muchas librerías precargadas, por lo que solo es necesario descargar algunas pocas al momento de ejecutar el Chatbot, además que se obtiene y configuran en el entorno los códigos del Chatbot al obtenerlos automáticamente del código almacenado en GitHub, lo cual se hace únicamente oprimiendo un botón para ejecutar esa parte de código, y posterior mente oprimir otro botón para ejecutar el Chatbot e iniciar una conversación. Con esto se obtiene gran accesibilidad al Chatbot, que el ingreso a la plataforma colaboratory con la configuración de nuestro sistema se realiza únicamente ingresando a un link. Esto en conjunto trae una gran accesibilidad desde cualquier dispositivo y en cualquier momento en tanto se tenga conexión a internet, sin necesidad de mayor configuración.

El resultado concreto de funcionamiento del Chatbot en Google Colaboratory es favorable ya que se ejecuta con buen tiempo de respuesta, y en una interfaz amigable fuera de la estructura del Chatbot como tal, que es compartida entre todas las versiones, con la única desventaja encontrada hasta ahora de que no se puede ejecutar la versión con reconocimiento de voz.

4.1.2 | Ambiente local: Computadora

La ejecución del Chatbot, en un ambiente local como una computadora, ya sea PC o Laptop requiere de una configuración inicial que puede ser un poco laboriosa, y en algún punto un tanto confusa si no se tiene un contexto previo sobre este tipo de software como anaconda, manejadores de entornos de desarrollo y entornos de desarrollo como tal, y si se esta distanciado del ambiente de programación. Anaconda es un manejador de ambientes de desarrollo para Python y R, el cual tiene muchos recursos y es bastante flexible, la ventaja es que una vez teniéndolo instalado, no solo sirve para ejecutar el Chatbot, si no para muchos otras trabajos, y para que el usuario desarrolle los propios, o si apetece agregue características al Chatbot, que siempre pueden ser compartidas con nosotros al repositorio en GitHub. Una vez que se tiene configurado el ambiente de trabajo solo es cuestion de entrar al notebook de Python, asegurarse de en la función de ejecución `chatear('C:/Inserta_la_direccin_a_la_capeta_Chatbot')`, se debe poner la dirección desde la raiz del sistema hacia la carpeta descargada de github del Chatbot para que se configuren las direcciones tanto del script de la estructura del Chatbot como de las bases de datos de donde se toma la información. y con eso se puede ejecutar, posterior a eso cada vez que se quiera volver a ejecutar únicamente se ingresa al notebook de Python y se corre la función `chatear('C:/Direccion/...')`.

La principal ventaja de utilizar esta versión es que se puede tener un mayor control si se quieres agregar funcionalidades al sistema, y además que se pueden ejecutar tanto la versión con entrada escrita como la que cuenta con reconocimiento de voz.

4.1.3 | Ambiente local: Tarjeta de Desarrollo

El implementar el sistema en Raspberry Pi contiene ventajas de portabilidad, además que demuestra la capacidad que tendría el sistema de integración a una plataforma móvil completa como un asistente de casa inteligente al poder ser incluido en sistemas con ese procesamiento, con además la capacidad de que si se cuentan con los periféricos necesarios la posibilidad de utilizar tanto la versión con entrada de texto, como la que contiene reconocimiento de voz. La desventaja es que el utilizar software tan demandante como lo puede llegar a ser *Anaconda*, o en especifico para Raspberry el software *MiniConda* puede ser mas procesamiento del necesario, por lo cual se podría plantear una implementación con una ventana emergente al ejecutarse como script de python desde la terminal, con la integración de código al sistema para crear una interfaz gráfica de funcionamiento. y que sea provechoso en este tipo de tarjetas de desarrollo.

4.2 | Interacción con el ChatBot

Se trató de hacer que la interacción con el Chatbot fuera lo más natural posible, pero también cuidando que no se complicará demasiado el problema. No todos los nodos tienen la misma cantidad de posibilidades, en algunos puede haber gran cantidad de categorías, lo que proporciona una amplia gama de elecciones pero por el otro lado puede dificultar un poco la rápida elección de categoría, sobre todo en el caso hablado. En otros casos donde hay pocas elecciones la interacción puede ser más rápida, pero también limita las decisiones, pudiendo resultar aburrido. Por último hay otros nodos donde no hay categorías definidas exactamente, la elección es dejada a la imaginación del usuario y el ChatBot busca encontrar una conexión entre la frase escuchada y parte de su base de datos, esto muchas veces resulta en varios intentos de consulta que se deben realizar para encontrar una recomendación del agrado total para el usuario, por lo que una solución para este problema es ampliar dichas bases de datos, o bien intentar recuperar mas información o información mas valiosa para tener un mayor grado de efectividad de buenas recomendaciones conforme a las preferencias del usuario.

4.2.1 | Seguimiento y control de la conversación

El ChatBot es quien se encarga de intentar controlar la conversación, guiando el camino a seguir según las opciones contenidas en el sistema mediante las elecciones que va realizando el usuario. Si se escoge o dice algo que no esta dentro de sus bases de datos o dentro de su programación, pedirá que se vuelva a decir otra opción recordando las categorías que tiene disponibles para ese nivel de decisión, en dicho el momento. El usuario pasa por los nodos escogidos y a final el chatbot preguntará si desea continuar conversando o no. Si se decide continuar se regresa a la *pregunta 1*, donde se hace la elección entre entretenimiento, académico e información de Covid. En caso contrario, es decir si ya no se quiere continuar la conversación, el ChatBot se despide y termina su ejecución.

4.2.2 | Entrada y salida de información

4.2.2.1 | ChatBot Escrito

Algunas ventajas de la versión escrita a parte de lo ya mencionado acerca de las plataformas de implementación, es que MMN ChatBot espera la entrada conteniendo la respuesta para poder continuar, por lo que la presión por continuar avanzando en la conversación no es tanta, puedes detenerte a leer tranquilamente las opciones desplegadas

o la información obtenida, meditar tu respuesta, abrir los links y revisarlos con calma. La desventaja en este sentido de entrada escrito también presenta la desventaja de que al leer la información puedes saltarte algunas partes y no entender todo correctamente la ifnormación que el chatbot intenta obtener del usuario, con esto, a la hora de escribir la respuesta es más probable que se cometan faltas de ortografía, o se utilicen palabras no contenidas en el vocabulario de entendimiento del sistema y el Chatbot al no entenderlo vuelva a requerir la entrada y repetir la pregunta. Otra cosa a tomar en cuenta es que al desplegar el mapa interactivo,la plataforma de Colab puede congelarse momentáneamente y parece no continuar la conversación, esto por lo pesado del procesamiento, pero se soluciona dando clic a la ejecución de la celda, esta continua recibiendo las entradas del usuario. Este problema no se tiene en las versiones en las demás versiones, o al correr la versión escrita en ambiente local.

4.2.2.2 | ChatBot Hablado

En el caso del chatbot la entrada y salida es un poco más natural, no es necesario detenerse a leer las instrucciones o las respuestas ya que el ChatBot las dice en voz alta, pero si necesitas re-consultar las opciones disponibles puedes leerlo en la pantalla otra vez, eso sí, hay un tiempo limite en el que Chatbot esta disponible para escuchar, si no escucha nada, vuelve a iniciar ese nodo de conversación, pero es importante fijarse estar hablando en el momento adecuado en el que el Chatbot escucha. Una de las ventajas de esta forma de interacción, es que es más natural responder en voz alta a sus preguntas, si se escucha claro se procesa la respuesta sin faltas de ortografía y el chatbot puede entender la respuesta más fácilmente. Por otro lado si no se escucha bien, por dicción del usuario o por mucho ruido en el ambiente de ejecución, esto puede hacer que se distorsionen las palabras y no lo entienda todo, esto dependerá más del ambiente y la cercanía al micrófono que se tenga. En las pruebas realizadas es muy bajo el índice de casos en el que se presenta esa situación de un incorrecto entendimiento de lo que se dice.

5

Conlusiones

En este proyecto se realizó la implementación de un ChatBot de recomendaciones por medio de árbol de decisión. Este sistema guía al usuario a través de la conversación dándole a elegir entre varias opciones para finalmente obtener una recomendación acorde a lo elegido. Para esto se uso el lenguaje de programación Python en archivos *.py* y *.ipynb* el cual contiene características que nos ayudan a realizar el manejo de datos de forma eficiente y a preservar una sintaxis clara de la estructura de funcionamiento general, por otro lado el uso de los notebooks de python nos permite tener una interfaz de uso del ChatBot eficiente y amigable donde se puede desplegar el historial de conversación, la información brindada y los enlaces en activo para re-dirigir al usuario a las actividades propuestas, así como desplegar entornos interactivos para visualización de información. El objetivo principal de este proyecto se cumplió al poder realizar un ChatBot recomendador por medio de árboles de decisión utilizando herramientas interactivas en su desarrollo. Esto desde su concepción teórica nos permitió tener un acercamiento a los conceptos vistos en clase y que optimizan en gran medida los esquemas generales de funcionamiento de sistemas dentro del área de Inteligencia Artificial. Con lo cual se ve de forma directa una aplicabilidad de estos conceptos a soluciones tecnológicas actuales y con gran popularidad como lo son los ChatBots. Durante el desarrollo se encontraron algunas limitantes como la accesibilidad a más variedad de datos, poca flexibilidad en el entendimiento general del lenguaje y limitaciones específicas en la implementación en línea para poder ejecutar la versión con reconocimiento de voz. Debido a esto la utilidad del sistema queda restringida a una actividad muy especifica que es la capacidad de dar recomendaciones y se tienen limitantes para tener na conversación más general con el usuario. Respecto al objetivo de accesibilidad, se puede decir que se satisfizo por completo dado que se tiene funcionando en línea y abierto a todo usuario en la

red. De acuerdo a las pruebas realizadas la tasa de éxito al obtener una recomendación es bastante buena, sin embargo en algunos casos la tasa de éxito en relación a brindar recomendaciones que sean del total agrado de los usuarios es completamente perfectible. Esto es posible lograrlo aumentando el contenido de las bases de datos y aumentando la complejidad de la red con nuevos nodos donde se obtenga información más valiosa acerca de las preferencias del usuario. Esto se resume a que se obtuvo un desarrollo con gran accesibilidad, facilidad de uso y con una construcción técnica robusta incluso cuando se tienen condiciones de ingreso de datos de manera oral.

5.1 | Posibles extensiones del proyecto

Como uno de los resultados obtenidos en este proyecto, destaca el surgimiento de una idea que es ampliamente extendible y que sirve como base para el desarrollo de sistemas más complejos dentro de esta área y que podrían conducir hacía el desarrollo de un producto en el que se incluya la funcionalidad de un ChatBot de recomendaciones como lo puede ser en los sistemas de streaming de películas y vídeos sugiriendo al usuario filmes en específico, en una tienda de ropa sugiriendo prendas de acuerdo a las preferencias del cliente, o como sistema de control de casa inteligente. Sin embargo el esquema propuesto hasta ahora puede ser robustecido con métodos de procesamiento del lenguaje natural que nos brinden mayor flexibilidad a la hora de interactuar con el usuario. Incluso migrar a sistemas basados en redes neuronales para los cuales también ya se han desarrollado entornos de trabajo que facilitan el uso de estos métodos para procesamiento del lenguaje. Estas ideas a su vez pueden ser extendidas y optimizadas para su desarrollo en dispositivos de procesamiento móvil en los cuales se agreguen funcionalidades de asistencia y automatización de tareas para distintos ambientes en los que el usuario lo requiera. Así como para la mejora de la interacción humano-máquina con dispositivos de uso cotidiano.

5.2 | Comentarios Finales

Con la realización de este proyecto, quedamos satisfechos con los resultados obtenidos dadas las condiciones y herramientas utilizadas, con la experimentación realizada, el cumplimiento de todos los objetivos y la experiencia de desarrollo adquirido. Logramos fortalecer nuestras habilidades de desarrollo específicamente en Python y nuestra experiencia acerca de la integración de herramientas en línea para la implementación de este tipo de sistemas. Dada la complejidad de las acciones que realiza el sistema fueron utilizadas librerías poco comunes como:

- Covid
- Deep Translator
- Speech Recognition
- PyAudio
- Pyttsx3
- Plotly
- PyCountry

A

Codigo de MMN ChatBot con Entrada Escrita

■ **Script de construcción de ChatBot.**

```
# -*- coding: utf-8 -*-
"""MMN Chatbot.ipynb

En este proyecto colaboraron:
- Natalia Sanchez Patino, github: @Natalia-SP
- Mario Rosas Otero, github: @Mariuki
- Mario Velazquez Vilchiz, github: @mvvazta

"""
### Librerias ###

import string
from covid import Covid
import random
import nltk
import pandas as pd
import numpy as np
import textwrap
import cv2
import sys
import unicodedata
from deep_translator import GoogleTranslator
import pycountry
import plotly.express as px
import wget
import os

```

```
28 """#Bases de Datos"""
29
30 Hombres = pd.read_csv('BasesDeDatos/nombreshombres .csv')
31 Mujeres = pd.read_csv('BasesDeDatos/nombresmujeres.csv')
32 Hombres = list(Hombres.iloc[:,0])
33 Mujeres = list(Mujeres.iloc[:,0])
34 Nombres = Hombres + Mujeres
35 Musica = pd.read_csv('BasesDeDatos/Music.csv')
36 Musica = pd.DataFrame(Musica)
37 categorias_musica = list(pd.unique(Musica['terms']))
38 Videos = pd.read_csv('BasesDeDatos/YTVideos.csv')
39 Videos = pd.DataFrame(Videos)
40 categorias_videos = list(pd.unique(Videos['category']))
41 Libros = pd.read_csv('BasesDeDatos/booksdataset.csv')
42 Libros = pd.DataFrame(Libros)
43 categorias_libros = list(pd.unique(Libros['category']))
44 Wiki = pd.read_csv('BasesDeDatos/WIKI.csv')
45 Wikis = pd.DataFrame(Wiki)
46 name_wikis = list(pd.unique(Wikis['Name']))
47 categorias_wikis = list(pd.unique(Wikis['Vertical1']))
48 Artic = pd.read_csv('BasesDeDatos/ArxivDataClean.csv')
49 Artic = pd.DataFrame(Artic)
50 VJ = pd.read_csv('BasesDeDatos/VGClean.csv')
51 VJ = pd.DataFrame(VJ)
52 categorias_vj = list(pd.unique(VJ['Genre']))
53 categorias2_vj = list(pd.unique(VJ['Platform']))
54 Netflix = pd.read_csv('BasesDeDatos/netflix_titlesClean.csv')
55 Netflix_p = pd.DataFrame(Netflix[Netflix['type']=='Movie'])
56 categorias_netp = list(pd.unique(Netflix_p['listed_in']))
57 Netflix_s = pd.DataFrame(Netflix[Netflix['type']=='TV Show'])
58 categorias_nets = list(pd.unique(Netflix_s['listed_in']))
59 Type_netflix = list(pd.unique(Netflix['type']))
60 Inv = pd.read_csv('BasesDeDatos/InvestigadoresSNIClean.csv')
61 Inv = pd.DataFrame(Inv)
62 categorias_inv = list(pd.unique(Inv['Area del Conocimiento']))
63 Area_inv = list(pd.unique(Inv['Area del Conocimiento']))
64
65
66 """# Listas de palabras frases y categorias"""
67
68 OP_ENTRETENIMIENTO = ["videos", "peliculas", "series", "musica", "libros","
       videojuegos","juegos"]
69 OP_ACADEMICO = ["articulo", "investigador", "investigadores", "articulos", "
       definiciones"]
70
```

```
71 SALUDOS_IN = ['Hola! Soy MMN Bot, mi especialidad es dar recomendaciones! Que
       tal va tu dia?', 'Hola! Que tal te sientes hoy?', 'Que onda, soy MMN Bot!
       Como te llamas?']
72 SALUDOS = ['hello', 'hi', 'hey', 'hola', 'welcome', 'bonjour', 'greetings', '
       que onda', 'holi']
73 SALUDOS_RESP = ["Hola, es cool hablar contigo!", 'Gusto en conocerte!', "Hey -
        Vamos a platicar un poco!"]
74
75 PREGUNTA_1 = ["Que quisieras que te recomendara, tengo la seccion de
       entretenimiento, academico y covid", "Muy bien, continuemos! Buscas algo
       academico, de entretenimiento o informacion sobre Covid?", "Me caes bien,
       puedo recomendarte algo academico, algo de entretenimiento o de Covid,
       cual prefieres?", "Sos la ostia, tengo para vosotros algo de
       entretenimiento, de covid o algo academico, elige..."]
76
77 LEER_NOMBRES = Nombres
78 DECIR_NOMBRES = ['gusto en conocerte, vamoa platicar :D', 'esta bien curado tu
        nombre, es un gusto.', ", ese nombre mola!, es un gusto conocerte.",'
       Gusto en conocerte!' , "Hey - Vamos a platicar un poco!"]
79
80 LEER_MUSICA = categorias_musica
81
82
83 LEER_LIBROS = ['no','poco', 'corto', 'medianito', 'menos', 'mucho', 'bastante'
       , 'largo', 'encanta']
84
85 NOMBRES_LIBROS = categorias_libros
86
87 LEER_VIDEOS = ['entretenimiento', 'peliculas', 'estilo', 'comedia', '
       tecnologia', 'blogs', 'deportes','activismo', 'noticias', 'gaming', '
       educacion', 'animales', 'autos', 'viajes', 'ciencia']
88
89 NOMBRES_VIDEOS = dict(zip(LEER_VIDEOS, categorias_videos))
90 NOMBRES_VIDEOS['ciencia'] = NOMBRES_VIDEOS['tecnologia']
91
92 LEER_INV = ['fisica','matematicas','tierra','biologia','quimica', 'medicina',
       'salud', 'humanidades','conducta', 'sociales', 'biotecnologia','
       agropecuarias','ingenierias']
93 DIC_INV = {'fisica': categorias_inv[4], 'matematicas': categorias_inv[4], '
       tierra': categorias_inv[4], 'biologia':categorias_inv[1], 'quimica':
       categorias_inv[1], 'medicina':categorias_inv[3], 'salud': categorias_inv
       [3], 'humanidades':categorias_inv[6], 'conducta':categorias_inv[6], '
       sociales':categorias_inv[2], 'biotecnologia':categorias_inv[0], '
       agropecuarias':categorias_inv[0], 'ingenierias':categorias_inv[5]}
94
```

```
95  LEER_PELIS = ['documentales','accion','comedia','palomera','drama', 'terror',
        'clasicos', 'ficcion','infantil']
96  DIC_PELIS = {'documentales': [categorias_netp[x] for x in [0,9,26,27]], '
        accion':[categorias_netp[x] for x in [1,2,16,18,24,32]], 'comedia':[
        categorias_netp[x] for x in [3,4,6,11,13,14,28,31,34]],'palomera':[
        categorias_netp[x] for x in [5]], 'drama':[categorias_netp[x] for x in
        [7,8,15,21,23,25,33]],'terror':[categorias_netp[x] for x in [10,19,20]],'
        clasicos':[categorias_netp[x] for x in [12]],'ficcion':[categorias_netp[x]
         for x in [17]],'infantil':[categorias_netp[x] for x in [22,29,30]]}
97
98  LEER_SERIES = ['crimen','novela','infantil','documentales','clasicos', '
        reality']
99  DIC_SERIES = {'crimen': [categorias_nets[x] for x in [0,5,7,9,12,14,19]], '
        novela':[categorias_nets[x] for x in [1,3,4,8,11,16,15]], 'infantil':[
        categorias_nets[x] for x in [2,18]],'documentales':[categorias_nets[x] for
         x in [6,17]], 'clasicos':[categorias_nets[x] for x in [10,13]],'reality'
        :[categorias_nets[x] for x in [20,21]]}
100
101 LEER_VJ_P = ['xbox','360','one','play','station','playstation', 'wii', 'psp',
        'computadora', 'compu', 'pc']
102 DIC_VJ_P = {'xbox': [categorias2_vj[x] for x in [4,13,17]], '360': [
        categorias2_vj[x] for x in [4,13,17]],'one': [categorias2_vj[x] for x in
        [4,13,17]], 'playstation':[categorias2_vj[x] for x in [5,6,10,12,16]],'
        play':[categorias2_vj[x] for x in [5,6,10,12,16]],'station':[
        categorias2_vj[x] for x in [5,6,10,12,16]],'psp':[categorias2_vj[x] for x
        in [5,6,10,12,16]], 'wii':[categorias2_vj[x] for x in [0,19]],'computadora
        ':[categorias2_vj[x] for x in [14]],'compu':[categorias2_vj[x] for x in
        [14]],'pc':[categorias2_vj[x] for x in [14]]}
103 LEER_VJ_G = ['deportes','plataforma','carreras','roles','rompecabezas','
        variado', 'disparos', 'simulacion', 'accion', 'peleas', 'aventura', "
        estrategia"]
104 DIC_VJ_G = dict(zip(LEER_VJ_G, categorias_vj))
105
106 LEER_CATEGORIAS = ['libros', 'libro', 'musica', 'videos', 'video', 'si', 'leer
        ']
107 DIC_LIBROS = {'no': 'short', 'poco': 'short', 'corto': 'short','medianito': '
        medium','mediano': 'medium','menos': 'medium', 'mucho': 'large', 'bastante
        ': 'large','encanta': 'large' ,'largo': 'large' }
108 LEER_COVID = ['cuarentena', 'covid', 'coronavirus', 'encerramiento', 'd 19', '
        sars', 'corona']
109
110
111 LEER_COMPU = ['python', 'codigo', 'computadora', 'algoritmo', ]
112 DECIR_COMPU = ["Python es de lo que estoy hecho.", \
113           "Sabias que estoy hecho con codigo!?", \
```

```
114             "Las computadoras son magicas", \
115             "Crees que podria pasar el Test de Turing?"]
116
117 LEER_CIENT = ['turing', 'hopper', 'neumann', 'lovelace']
118 DECIR_CIENT = ['fue asombroso!', 'hizo muchas cosas importantes!', 'es alguien
        del que que valdria la pena saber mas :).']
119 NOMBRES_CIENT = {'turing': 'Alan', 'hopper': 'Grace', 'neumann': 'John von', '
        lovelace': 'Ada'}
120
121 LEER_BROMAS = ['divertido', 'gracioso', 'ja', 'jaja', 'jajaja', 'xD']
122 DECIR_BROMAS = ['ja!', 'jajaja!', 'XD', 'lol']
123
124 LEER_NEGACIONES = ['matlab', 'java', 'C++']
125 DECIR_NEGACIONES = ["No, lo siento. :(, No me gustaria hablar por ahora de
        eso."]
126
127 NEGATIVAS = ['no', "no no", 'nop', 'nunca', "negativo", "ninguno"]
128
129 DESCONOCIDO = ['Bien.', 'Okay', 'Mm?', 'Si!', 'bien...', 'am', 'Hum']
130 CHATEAR = ['Que te gustaria hacer ahora?, puedo recomedarte algo de musica,
        libros o algun video enretenido, Cual te gustaria?', 'Veamos, Que tipo de
        musica te gusta?', 'Quieres algo para relajarte?', 'Puedo buscar algo de
        buena musica para ti,Que genero te gusta?', 'Tengo algunos videos
        entretenidos!, escoge una categoria :D','Te gustan los videos? Tengo de
        diferentes categorias', 'ademas, tengo aqui algunos de mis libros
        favoritos, te gusta leer mucho, mas o menos, o solo un poco?',
131             'Sobre que deberia buscar?']
132
133 RESP_PREG = "Soy demasiado timido para a responder eso, jeje. De que otra cosa
        te gustaria una reomendacion?"
134
135 ### Funciones ###
136
137 def es_pregunta(entrada):
138   for i in entrada:
139     if i == '?':
140       salida = True
141     else:
142       salida = False
143   return salida
144
145 def quitar_acentos(string):
146     acentos = set(map(unicodedata.lookup, ('COMBINING ACUTE ACCENT', 'COMBINING
          GRAVE ACCENT', 'COMBINING TILDE')))
```

```
147     chars = [c for c in unicodedata.normalize('NFD', string) if c not in
            acentos]
148     return unicodedata.normalize('NFC', ''.join(chars))
149
150 def remover_puntuacion(entrada):
151   out_string = ""
152   for i in entrada:
153     if i not in string.punctuation:
154       out_string += i
155   return out_string
156
157 def preparar_texto(entrada):
158   temp_string = entrada.lower()
159   temp_string = remover_puntuacion(temp_string)
160   temp_string = quitar_acentos(temp_string)
161   lista_salida = temp_string.split()
162   return lista_salida
163
164 def responder_echo(entrada, numero_bromas,espaciador):
165   if entrada != None:
166     echo_salida = (entrada + espaciador) * numero_bromas
167   else:
168     echo_salida = None
169   return echo_salida
170
171 def selector(lista_entrada, checar_lista, regresar_lista):
172   salida = None
173   for i in lista_entrada:
174     if i in checar_lista:
175       salida = random.choice(regresar_lista)
176       break
177   return salida
178
179 def concatenar_string(string1, string2, separador):
180   salida = string1 + separador + string2
181   return salida
182
183 def lista_a_cadena(lista_entrada, separador):
184   salida = lista_entrada[0]
185   for i in lista_entrada[1:]:
186     salida = concatenar_string(salida, i, separador)
187   return salida
188
189 def esta_en_lista(lista_uno, lista_dos): #Checar si cualqueir elemento esta en
        dos listas.
```

```
190
191     for elemento in lista_uno:
192         if elemento in lista_dos:
193             return True
194     return False
195
196 def encontrar_en_lista(lista_uno, lista_dos): # Find and return an element
        from list_one that is in list_two, or None otherwise.
197     for elemento in lista_uno:
198         if elemento in lista_dos:
199             return elemento
200     return None
201
202 def terminar_chat(lista_entrada):
203   if encontrar_en_lista(lista_entrada, ["no", "adios","nelson","bye","chao","
        vemos","nel"]):
204     salida = True
205   else:
206     salida = False
207   return salida
208
209 def contar_puntos(entrada):
210   p = 0
211   h = []
212   for i in entrada:
213     o = i.count('.')
214     if o == 1:
215       p += 1
216       if p == 5:
217         break
218     h.append(i)
219   h.append('.')
220   return h
221
222 def codigo_pais(nombre):
223     try:
224         return pycountry.countries.lookup(nombre).alpha_3
225     except:
226         return None
227
228 def creargrafica():
229     wget.download("https://raw.githubusercontent.com/CSSEGISandData/COVID-19/
            master/csse_covid_19_data/csse_covid_19_time_series/
            time_series_covid19_confirmed_global.csv", bar=None)
230     df_confirm = pd.read_csv('time_series_covid19_confirmed_global.csv')
```

```
231     df_confirm = df_confirm.drop(columns=['Province/State','Lat', 'Long'])
232     df_confirm = df_confirm.groupby('Country/Region').agg('sum')
233     date_list = list(df_confirm.columns)
234     df_confirm['country'] = df_confirm.index
235     df_confirm['iso_alpha_3'] = df_confirm['country'].apply(codigo_pais)
236     df_long = pd.melt(df_confirm, id_vars=['country','iso_alpha_3'], value_vars
            =date_list)
237     fig = px.choropleth(df_long, # Input Dataframe
238                     locations="iso_alpha_3", # identify country code column
239                     color="value", # identify representing column
240                     hover_name="country", # identify hover name
241                     animation_frame="variable", # identify date column
242                     projection="natural earth", # select projection
243                     color_continuous_scale = 'rainbow', # select prefer color
                            scale
244                     range_color=[0,50000] # select range of dataset
245                     )
246     os.remove("time_series_covid19_confirmed_global.csv")
247     return fig.show()
248
249 def leer_mensaje(tunombre="INPUT", w=False):
250     texto_entrada = None
251     mensaje = None
252     try:
253         texto_entrada = input(chr(27)+"[1;30m"+str(tunombre) +': \t')
254         # print(chr(27)+"[1;30m"+str(tunombre) +': \t' + str(texto_entrada))
255         if w:
256             mensaje = texto_entrada.split() #w se usa para Wikipedia
257             mensaje = [mensaje.capitalize() for mensaje in mensaje]
258         else:
259             mensaje = preparar_texto(texto_entrada)
260     except:
261         print(chr(27)+"[1;31m"+"CHATBOT: No he podido entenderte, intenta de
                nuevo")
262
263         mensaje = leer_mensaje(tunombre)
264     return mensaje
265
266
267 ### Caso 2 ###
268 def videos(tunombre):
269     msg_salida = None
270     msg_salida = random.choice(["Que tipos de videos te gustarian?, tengo de:\n
            ", "Genial!, tengo estas categorias:\n", "Muy bien, revisare mi
            coleccion favorita de videos, podriamos empezar por: \n"])
```

```
271     opciones_videos = " - Comedia\n - Tecnologia\n - Peliculas\n - Estilo\n -
            Entretenimiento\n - Blogs\n - Deportes\n - Activismo\n - Noticias\n -
            Gaiming\n - Educacion\n - Animales\n - Autos\n - Viajes\n - Ciencia"
272
273     print(chr(27)+"[1;34m"+'CHATBOT:')
274
275     for i in textwrap.wrap(str(msg_salida), 130):
276       print(chr(27)+"[1;34m"+ i)
277     print("\n"+chr(27)+"[1;34m"+'CHATBOT:\n'+opciones_videos)
278     msg = leer_mensaje(tunombre)
279     name = encontrar_en_lista(list(msg), LEER_VIDEOS)
280     if name:
281         ran = np.random.randint(0,len(Videos[Videos['category']==NOMBRES_VIDEOS
                [name]]))
282         title = Videos[Videos['category']==NOMBRES_VIDEOS[name]][['title', '
                video_id']]
283         msg_salida = (
284             'Si te gusta {} yo te recomendaria este video "{}" , lo puedes ver
                    en https://www.youtube.com/watch?v={}'.format(name,
285                 title.iloc[ran][0], title.iloc[ran][1]))
286     else:
287         msg_salida = "Hum... creo que no capte algo de lo que dijiste, podrias
                repetirlo?"
288         print(chr(27)+"[1;34m"+'CHATBOT:')
289
290         for i in textwrap.wrap(str(msg_salida), 130):
291           print(chr(27)+"[1;34m"+ i)
292         videos(tunombre)
293         msg_salida = []
294
295     print(chr(27)+"[1;34m"+'CHATBOT:')
296
297     for i in textwrap.wrap(str(msg_salida), 130):
298       print(chr(27)+"[1;34m"+ i)
299
300
301     return
302
303 def peliculas(tunombre):
304     msg_salida = None
305     msg_salida = random.choice(["Que clase de peliculas te gustan?, tengo\n", "
            Genial!, tengo estas categorias\n", "Muy bien, entretenimiento,
            podriamos empezar por: \n"])
306     opciones_peliculas = " - documentales\n - accion\n - comedia\n - palomera\n
             - drama\n - terror\n - clasicos\n - ficcion\n - infantil"
```

```
307     # msg_salida = msg_salida + opciones_peliculas
308     print(chr(27)+"[1;34m"+'CHATBOT:')
309
310     for i in textwrap.wrap(str(msg_salida), 130):
311       print(chr(27)+"[1;34m"+ i)
312     print("\n"+chr(27)+"[1;34m"+'CHATBOT:\n'+opciones_peliculas)
313     msg = leer_mensaje(tunombre)
314     name = encontrar_en_lista(msg, LEER_PELIS)
315     if name:
316         select = random.choice(DIC_PELIS[name])
317         ran = np.random.randint(0,len(Netflix_p[Netflix_p['listed_in']==select
                ]))
318         datos = Netflix_p[Netflix_p['listed_in']==select][['title', 'duration',
                'description']].iloc[ran]
319         msg_salida = (
320             'Si te gusta la categoria de {} yo te recomendaria la pelicula "{}",
                     que dura {}, trata de: {}'.format(name,
321                 datos[0], GoogleTranslator(source='auto', target='es').translate
                        (datos[1]), GoogleTranslator(source='auto', target='es').
                        translate(datos[2])))
322     else:
323         msg_salida = "Hum... creo que no capte algo de lo que dijiste, podrias
                repetirlo?"
324         print(chr(27)+"[1;34m"+'CHATBOT:')
325
326         for i in textwrap.wrap(str(msg_salida), 130):
327           print(chr(27)+"[1;34m"+ i)
328         peliculas(tunombre)
329         msg_salida = []
330
331     print(chr(27)+"[1;34m"+'CHATBOT:')
332
333     for i in textwrap.wrap(str(msg_salida), 130):
334       print(chr(27)+"[1;34m"+ i)
335
336
337     return
338
339 def series(tunombre):
340     msg_salida = None
341     msg_salida = random.choice(["Que tipo de series te gustan?, tengo\n", "
            Genial!, tengo estas categorias\n", "Muy bien, series, podriamos
            empezar por: \n"])
342     categorias_series = " - Crimen\n - Novela\n - Infantil\n - Documentales\n -
             Clasicos\n - Reality Shows"
```

```
343     # msg_salida = msg_salida + categorias_entretenimiento
344     print(chr(27)+"[1;34m"+'CHATBOT:')
345 
346     for i in textwrap.wrap(str(msg_salida), 130):
347       print(chr(27)+"[1;34m"+ i)
348     print("\n"+chr(27)+"[1;34m"+'CHATBOT:\n'+categorias_series)
349     msg = leer_mensaje(tunombre)
350     name = encontrar_en_lista(msg, LEER_SERIES)
351     if name:
352         select = random.choice(DIC_SERIES[name])
353         ran = np.random.randint(0,len(Netflix_s[Netflix_s['listed_in']==select
                ]))
354         datos = Netflix_s[Netflix_s['listed_in']==select][['title', 'duration',
                'description']].iloc[ran]
355         msg_salida = (
356     'Si te gusta la categoria de {} yo te recomendaria la serie "{}", que tiene
            {}, trata de: {}'.format(name,
357         datos[0], GoogleTranslator(source='auto', target='es').translate(datos
                [1]), GoogleTranslator(source='auto', target='es').translate(datos
                [2])))
358     else:
359         msg_salida = "Hum... creo que no capte algo de lo que dijiste, podrias
                repetirlo?"
360         print(chr(27)+"[1;34m"+'CHATBOT:')
361 
362         for i in textwrap.wrap(str(msg_salida), 130):
363           print(chr(27)+"[1;34m"+ i)
364         series(tunombre)
365         msg_salida = []
366 
367     print(chr(27)+"[1;34m"+'CHATBOT:')
368 
369     for i in textwrap.wrap(str(msg_salida), 130):
370       print(chr(27)+"[1;34m"+ i)
371 
372 
373     return
374 
375 def musica(tunombre):
376     msg_salida = None
377     msg_salida = random.choice(["Que genero de musica prefieres?\n", "Genial!,
            Que musica te gusta?\n", "La musica es genial!, podriamos empezar por
            decirme tu genero favorito \n"])
378 
379     print(chr(27)+"[1;34m"+'CHATBOT:')
```

```
380
381     for i in textwrap.wrap(str(msg_salida), 130):
382       print(chr(27)+"[1;34m"+ i)
383
384     msg = leer_mensaje(tunombre)
385     name = encontrar_en_lista(msg, LEER_MUSICA)
386     if name:
387         ran = np.random.randint(0,len(Musica[Musica['terms']==name]))
388         title = Musica[Musica['terms']==name][['release.name', 'artist.name']]
389         msg_salida = (
390             'Si te gusta el {} te recomiendo esta cancion "{}" de {}'.format(
                    name,
391                 title.iloc[ran][0], title.iloc[ran][1]))
392     else:
393         msg_salida = "Hum... creo que no capte algo de lo que dijiste, podrias
                repetirlo?"
394         print(chr(27)+"[1;34m"+'CHATBOT:')
395
396         for i in textwrap.wrap(str(msg_salida), 130):
397           print(chr(27)+"[1;34m"+ i)
398         musica(tunombre)
399         msg_salida = []
400
401     print(chr(27)+"[1;34m"+'CHATBOT:')
402
403     for i in textwrap.wrap(str(msg_salida), 130):
404       print(chr(27)+"[1;34m"+ i)
405
406
407     return
408
409 def libros(tunombre):
410     msg_salida = None
411     msg_salida = random.choice(["Que tanto te gusta leer?, mucho, poco?\n", "
            Genial!, podria sugerirte un libro corto, medianito o algo largo.\n",
            "Muy bien, libros, que tan grandes? mucho, mas o menos, poco...: \n"])
412
413     print(chr(27)+"[1;34m"+'CHATBOT:')
414
415     for i in textwrap.wrap(str(msg_salida), 130):
416       print(chr(27)+"[1;34m"+ i)
417
418     msg = leer_mensaje(tunombre)
419     name = encontrar_en_lista(msg, LEER_LIBROS)
420     if name:
```

```
421         ran = np.random.randint(0,len(Libros[Libros['category']==DIC_LIBROS[
                name]]))
422         title = Libros[Libros['category']==DIC_LIBROS[name]][['title', 'authors
                ', 'num_pages']]
423         msg_salida = (
424          'Este libro "{}" suena bien para ti, fue escrito por {} y tiene {}
                 paginas.'.format(
425             title.iloc[ran][0], title.iloc[ran][1], title.iloc[ran][2]))
426     else:
427         msg_salida = "Hum... creo que no capte algo de lo que dijiste, podrias
                repetirlo?"
428         print(chr(27)+"[1;34m"+'CHATBOT:')
429 
430         for i in textwrap.wrap(str(msg_salida), 130):
431           print(chr(27)+"[1;34m"+ i)
432         libros(tunombre)
433         msg_salida = []
434 
435     print(chr(27)+"[1;34m"+'CHATBOT:')
436 
437     for i in textwrap.wrap(str(msg_salida), 130):
438       print(chr(27)+"[1;34m"+ i)
439 
440 
441     return
442 
443 def juegos(tunombre):
444     msg_salida = None
445     msg_salida = random.choice(["Que clase de videojuegos te gustan?, que
            plataforma usas?\n", "Genial!, dime una categoria y plataforma.\n", "
            Muy bien, videojuegos, de que tipo, que consola?: \n"])
446     categorias_videojuegos = " Categorias:\t Consolas:\n - Accion\t + Xbox\n -
            Aventuras\t + PlayStation\n - Carreras\t + Wii\n - Deportes\t +
            Computadora\n - Disparos\n - Estrategia\n - Peleas\n - Plataforma\n -
            Roles\n - Rompecabezas\n - Simulacion\n - Variado"
447     # msg_salida = msg_salida + categorias_videojuegos
448     print(chr(27)+"[1;34m"+'CHATBOT:')
449 
450     for i in textwrap.wrap(str(msg_salida), 130):
451       print(chr(27)+"[1;34m"+ i)
452     print("\n"+chr(27)+"[1;34m"+'CHATBOT:\n'+categorias_videojuegos)
453     msg = leer_mensaje(tunombre)
454     keyp = encontrar_en_lista(msg, LEER_VJ_P)
455     keyg = encontrar_en_lista(msg, LEER_VJ_G)
456     if keyp and keyg:
```

```
457         Keyp = random.choice(DIC_VJ_P[keyp])
458         Keyg = DIC_VJ_G[keyg]
459         VJG = VJ[VJ['Genre'] == Keyg]
460         VJG = VJG[VJG['Platform'] == Keyp]
461         ran = np.random.randint(0,len(VJG))
462         datos = VJG[['Name','Genre', 'Platform']].iloc[ran]
463         msg_salida = (
464             'Si te gusta la categoria de {} yo te recomendaria "{}", para {}.'.
                format(keyg,
465                datos[0], datos[2]))
466     else:
467         msg_salida = "Hum... creo que no capte algo de lo que dijiste, podrias
                repetirlo?"
468         print(chr(27)+"[1;34m"+'CHATBOT:')
469 
470         for i in textwrap.wrap(str(msg_salida), 130):
471           print(chr(27)+"[1;34m"+ i)
472         juegos(tunombre)
473         msg_salida = []
474 
475     print(chr(27)+"[1;34m"+'CHATBOT:')
476 
477     for i in textwrap.wrap(str(msg_salida), 130):
478       print(chr(27)+"[1;34m"+ i)
479 
480 
481     return
482 
483 def articulos(tunombre):
484     msg_salida = None
485     msg_salida = random.choice(["Me agrada que quieras descubrir conocimiento,
            di una palabra clave (en ingles)\n", "Genial! dime una palabra clave (
            en ingles), para encontrar uno interesante\n", "Muy bien, busquemos
            uno interesante, dime una palabra clave que podria interesarte: \n"])
486     print(chr(27)+"[1;34m"+'CHATBOT:')
487 
488     for i in textwrap.wrap(str(msg_salida), 130):
489       print(chr(27)+"[1;34m"+ i)
490 
491     subartic = []
492 
493     w = leer_mensaje(tunombre,w=True)
494     print(chr(27)+"[1;34m"+'Buscando alguna coincidencia...')
495     w = random.choice(w)
496     try:
```

```
497         subArtic = Artic[Artic['title'].str.contains(w)]
498         # print(subArtic)
499 
500         ran = np.random.randint(0,len(subArtic))
501         title = Artic[['title']].iloc[ran][0]
502         id = Artic[['id']].iloc[ran][0]
503         msg_salida = (
504             'Un articulo relacionado a {} que encontre para ti: {}'.format(w,
                    GoogleTranslator(source='auto', target='es').translate(title))
                    +
505              ' Puedes leerlo completo en: https://arxiv.org/abs/{}'.format(id))
506     except:
507         msg_salida = "Hum... creo que no capte algo de lo que dijiste, podrias
                repetirlo?"
508         print(chr(27)+"[1;34m"+'CHATBOT:')
509 
510         for i in textwrap.wrap(str(msg_salida), 130):
511           print(chr(27)+"[1;34m"+ i)
512         articulos(tunombre)
513         msg_salida = []
514 
515     print(chr(27)+"[1;34m"+'CHATBOT:')
516 
517     for i in textwrap.wrap(str(msg_salida), 130):
518       print(chr(27)+"[1;34m"+ i)
519 
520     return
521 
522 def wikis(tunombre):
523     msg_salida = None
524     msg_salida = random.choice(["Que te gustaria saber?, preguntame algun
            concepto\n", "Genial!, te interesa saber la definicion de algo en
            particular?\n", "Muy bien, podriamos empezar por algo que quieras
            saber... \n"])
525     print(chr(27)+"[1;34m"+'CHATBOT:')
526 
527     for i in textwrap.wrap(str(msg_salida), 130):
528       print(chr(27)+"[1;34m"+ i)
529 
530     w = leer_mensaje(tunombre,w=True)
531     w = GoogleTranslator(source='auto', target='en').translate(lista_a_cadena(w
            , ''))
532     w = w.split()
533     name = encontrar_en_lista(w, name_wikis)
534     if name:
```

```
535         ran = np.random.randint(0,len(Wikis[Wikis['Name']==name]))
536         title = Wikis[Wikis['Name']==name][['Name', 'WikiDescription','WikiUrl'
                ]]
537         msg_salida = (
538             'Aqui esta la definicion de {} que encontre para ti: {}'.format(
539                 GoogleTranslator(source='auto', target='es').translate(
                        title.iloc[ran][0]), GoogleTranslator(source='auto', target=
                        'es').translate(lista_a_cadena(contar_puntos(title.iloc[ran
                        ][1]), ''))) +
540              ' Puedes leer mas de ello en: {}'.format(
541                 title.iloc[ran][2]))
542     else:
543         msg_salida = "Hum... creo que no capte algo de lo que dijiste, podrias
                repetirlo?"
544         print(chr(27)+"[1;34m"+'CHATBOT:')
545 
546         for i in textwrap.wrap(str(msg_salida), 130):
547           print(chr(27)+"[1;34m"+ i)
548         wikis(tunombre)
549         msg_salida = []
550 
551     print(chr(27)+"[1;34m"+'CHATBOT:')
552 
553     for i in textwrap.wrap(str(msg_salida), 130):
554       print(chr(27)+"[1;34m"+ i)
555 
556     return
557 
558 def investigadores(tunombre):
559     msg_salida = None
560     msg_salida = random.choice(["Que area del conocimiento te agrada en este
            momento?, estas son:\n", "Genial!, las areas en las que podria
            encontrar a alguien son\n", "Muy bien, en que area estas interesado: \
            n"])
561     categorias_inv = " - Fisica, Matematicas y Ciencias de la Tierra\n -
            Biologia y Quimica\n - Medicina y Ciencias de la Salud\n - Humanidades
             y Ciencias de la Conducta\n - Ciencias Sociales\n - Biotecnologia y
            Ciencias Agropecuarias\n - Ingenierias"
562     # msg_salida = msg_salida + categorias_inv
563     print(chr(27)+"[1;34m"+'CHATBOT:')
564 
565     for i in textwrap.wrap(str(msg_salida), 130):
566       print(chr(27)+"[1;34m"+ i)
567     print("\n"+chr(27)+"[1;34m"+'CHATBOT:\n'+categorias_inv)
568     msg = leer_mensaje(tunombre)
```

```
569     name = encontrar_en_lista(msg, LEER_INV)
570     if name:
571         ran = np.random.randint(0,len(Inv[Inv['Area del Conocimiento']==DIC_INV
                [name]]))
572         datos = Inv[Inv['Area del Conocimiento']==DIC_INV[name]][['Nombre
                Completo', 'Area del Conocimiento', 'Institucion de Adscripcion']].
                iloc[ran]
573         msg_salida = (
574             'Si te gusta el area de {} yo te recomendaria contactar o buscar el
                    trabajo desarrollado por "{}", adscrito a {}'.format(datos[1],
                    datos[0], datos[2]))
575     else:
576         msg_salida = "Hum... creo que no capte algo de lo que dijiste, podrias
                repetirlo?"
577         print(chr(27)+"[1;34m"+'CHATBOT:')
578 
579         for i in textwrap.wrap(str(msg_salida), 130):
580           print(chr(27)+"[1;34m"+ i)
581         investigadores(tunombre)
582         msg_salida = []
583 
584     print(chr(27)+"[1;34m"+'CHATBOT:')
585 
586     for i in textwrap.wrap(str(msg_salida), 130):
587       print(chr(27)+"[1;34m"+ i)
588 
589 
590     return
591 
592 def covids(tunombre):
593 
594     msg_salida = None
595     msg_salida = random.choice(["Muy bien! Aqui hay algo de informacion sobre
            Covid. Cuida tu salud!\n Te dejo los datos actualizados sobre covid en
             Mexico\n Y un mapa interactivo de la evolucion de covid en el mundo."
            ])
596     print(chr(27)+"[1;34m"+'CHATBOT:')
597 
598     for i in textwrap.wrap(str(msg_salida), 130):
599       print(chr(27)+"[1;34m"+ i)
600     covid = Covid()
601     casos = covid.get_status_by_country_name(country_name='mexico')
602     print(chr(27)+"[1;31m"+'CHATBOT: Aqui hay un poco de informacion
            actualizada de Covid en Mexico: \n')
603     for x in casos:
```

```
604         print(chr(27)+"[1;31m"+ x, ':', casos[x])
605     creargrafica()
606     return
607 
608 ### Switchs ###
609 
610 def switcher_entretenimiento(key,tunombre):
611     switch_entretenimiento = {
612         "videos": videos,
613         "peliculas": peliculas,
614         "series": series,
615         "musica": musica,
616         "libros": libros,
617         "videojuegos": juegos,
618         "juegos": juegos
619     }
620     funcion = switch_entretenimiento.get(key)
621     return funcion(tunombre)
622 
623 def switcher_academico(key,tunombre):
624     switch_academico = {
625         "articulos": articulos,
626         "articulo": articulos,
627         "investigadores": investigadores,
628         "investigador": investigadores,
629         "definicion": wikis,
630         "definiciones": wikis,
631     }
632     funcion = switch_academico.get(key)
633     return funcion(tunombre)
634 
635 ### Casos 1###
636 def entretenimiento(tunombre):
637     msg_salida = None
638     msg_salida = random.choice(["Que te gustaria de entretenimiento?, tengo\n",
          "Genial!, tengo estas categorias\n", "Muy bien, entretenimiento,
         podriamos empezar por: \n"])
639     categorias_entretenimiento = " - Videos\n - Peliculas\n - Series\n - Musica
         \n - Libros\n - Videojuegos"
640     # msg_salida = msg_salida + categorias_entretenimiento
641     print(chr(27)+"[1;34m"+'CHATBOT:')
642 
643     for i in textwrap.wrap(str(msg_salida), 130):
644       print(chr(27)+"[1;34m"+ i)
645     print("\n"+chr(27)+"[1;34m"+'CHATBOT:\n'+categorias_entretenimiento)
```

```
646     msg = leer_mensaje(tunombre)
647     key = encontrar_en_lista(msg, OP_ENTRETENIMIENTO)
648     if key:
649         switcher_entretenimiento(key,tunombre)
650     else:
651         msg_salida = "Hum... creo que no capte algo de lo que dijiste, podrias
                repetirlo?"
652         print(chr(27)+"[1;34m"+'CHATBOT:')
653 
654         for i in textwrap.wrap(str(msg_salida), 130):
655           print(chr(27)+"[1;34m"+ i)
656         entretenimiento(tunombre)
657         msg_salida = []
658     return
659 
660 def academico(tunombre):
661     msg_salida = None
662     msg_salida = random.choice(["Tengo distintas recomendaciones academicas,
            algunas son:\n", "Genial!, tengo estas categorias\n", "Muy bien, el
            ambito academico, podriamos empezar por: \n"])
663     categorias_academico = " - Articulos\n - Investigadores\n - Definiciones"
664     # msg_salida = msg_salida + categorias_academico
665     print(chr(27)+"[1;34m"+'CHATBOT:')
666 
667     for i in textwrap.wrap(str(msg_salida), 130):
668       print(chr(27)+"[1;34m"+ i)
669     print("\n"+chr(27)+"[1;34m"+'CHATBOT:\n'+categorias_academico)
670     msg = leer_mensaje(tunombre)
671     key = encontrar_en_lista(msg, OP_ACADEMICO)
672     if key:
673         switcher_academico(key,tunombre)
674     else:
675         msg_salida = "Hum... creo que no capte algo de lo que dijiste, podrias
                repetirlo?"
676         print(chr(27)+"[1;34m"+'CHATBOT:')
677 
678         for i in textwrap.wrap(str(msg_salida), 130):
679           print(chr(27)+"[1;34m"+ i)
680         academico(tunombre)
681         msg_salida = []
682 
683     return
684 
685 def switcher_general(key,tunombre):
686     switch_general = {
```

```
687         "entretenimiento": entretenimiento,
688         "academico": academico,
689         "covid": covids
690     }
691 
692     funcion = switch_general.get(key)
693     return funcion(tunombre)
694 
695 def general(tunombre):
696     msg = leer_mensaje(tunombre)
697     primer_mensaje = encontrar_en_lista(msg, ["entretenimiento","academico","
            covid"])
698     if primer_mensaje:
699         switcher_general(primer_mensaje, tunombre)
700     else:
701         msg_salida = "Hum... creo que no capte algo de lo que dijiste, podrias
                repetirlo?"
702         print(chr(27)+"[1;34m"+'CHATBOT:')
703 
704         for i in textwrap.wrap(str(msg_salida), 130):
705           print(chr(27)+"[1;34m"+ i)
706         general(tunombre)
707         msg_salida = []
708     return
709 
710 def chatear():
711     """funcion principal para tener un chat."""
712     print(chr(27)+"[1;34m"+'Que tal! Soy tu amigo MMN Bot! Cual es tu nombre?:
            \n')
713 
714     chat = True
715     tunombre = None
716     try:
717         msg = input(chr(27)+"[1;30m"+"INPUT" +': \t')
718         # print(chr(27)+"[1;30m"+str(tunombre) +': \t' + str(texto_entrada))
719     except:
720         print(chr(27)+"[1;31m"+"No he podido entenderte, intenta de nuevo")
721 
722         msg = leer_mensaje(tunombre)
723     if tunombre != None:
724       print(chr(27)+"[1;30m"+str(tunombre) +': \t' + msg)
725     else:
726       # print(chr(27)+"[1;30m"+'INPUT : \t' + msg)
727       n = msg.upper() # n sirve para la funcion de nombres
728       n = n.split() # en lugar de la funcion preparar_texto
```

```
729     for i in n:
730       i = [i]
731       if esta_en_lista(i, LEER_NOMBRES):
732           tunombre = encontrar_en_lista(i, LEER_NOMBRES)
733           msg_salida =(lista_a_cadena([tunombre.capitalize(),
734                                        selector(i, LEER_NOMBRES, DECIR_NOMBRES)],
                                             ' '))
735   while chat:
736       msg = None
737       msg_salida = None
738       preg1 = random.choice(PREGUNTA_1)
739       print(chr(27)+"[1;34m"+'CHATBOT: \t'+ preg1)
740
741
742       general(tunombre)
743       msg_salida = "Genial, un gusto hablar contigo quieres continuar
              conversando?"
744       print(chr(27)+"[1;34m"+'CHATBOT: \t'+ msg_salida)
745
746       msg = leer_mensaje(tunombre)
747       if terminar_chat(msg):
748           msg_salida = 'Adios!'
749           print(chr(27)+"[1;34m"+'CHATBOT: \t'+ msg_salida)
750
751           chat = False
```

■ **Python Notebook ChatBot Escrito COLAB**

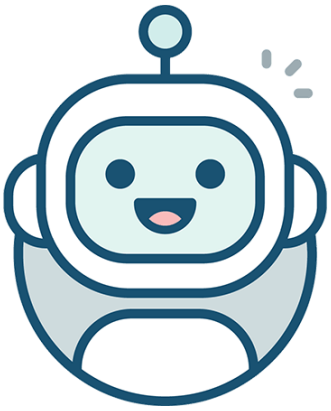
Librerias

Corre esta celda

In [1]:
```
!pip install covid
!pip install pycountry
!pip install plotly
!pip install wget
!pip install deep-translator
!git clone https://github.com/NM-Labs/ChatBot.git
```

Interactuar

Corre la celda de abajo para interactuar con MMNBot. Puedes interactuar con el de forma normal, solo intenta seguir la corriente.



In [2]:
```
#@title MMN Chatbot
#@markdown ![Esta es una imagen de ejemplo] \\
    (https://encrypted-tbn0.gstatic.com/images?q=tbn \\
    3AANd9GcRGbLPwoNkbvHyZMHYn5F-8gnr2rF7BPKXnFw&usqp=CAU)
%cd ChatBot
import chatESC as c
c.chatear()
```

B

Codigo de MMN ChatBot con Entrada Hablada

■ **Script de construcción de ChatBot.**

```
1
2  # -*- coding: utf-8 -*-
3  """MMN Chatbot.ipynb
4
5  En este proyecto colaboraron:
6  - Natalia Sanchez Patino, github: @Natalia-SP
7  - Mario Rosas Otero, github: @Mariuki
8  - Mario Velazquez Vilchiz, github: @mvvazta
9
10 """
11 ### Librerias ###
12
13 #!pip install covid
14
15 import string
16 from covid import Covid
17 import random
18 import nltk
19 import pandas as pd
20 import numpy as np
21 import textwrap
22 import cv2
23 import speech_recognition as SRG
24 import time
25 import sys
26 import unicodedata
27 import pyttsx3
```

```
28 from deep_translator import GoogleTranslator
29 import pycountry
30 import plotly.express as px
31 import wget
32 import os
33
34 st = SRG.Recognizer()
35
36 """#Bases de Datos"""
37 def getDatos(path="/"):
38     global LEER_NOMBRES, DECIR_NOMBRES, OP_ENTRETENIMIENTO, OP_ACADEMICO,
          SALUDOS_IN, SALUDOS, SALUDOS_RESP, PREGUNTA_1, LEER_MUSICA,
          LEER_LIBROS, NOMBRES_LIBROS, LEER_VIDEOS, NOMBRES_VIDEOS, LEER_INV,
          LEER_PELIS, DIC_PELIS, DIC_INV, LEER_SERIES, DIC_SERIES, LEER_VJ_P,
          DIC_VJ_P, LEER_VJ_G, DIC_VJ_G, LEER_CATEGORIAS, LEER_COVID, LEER_COMPU
          , DECIR_COMPU, LEER_CIENT, NOMBRES_CIENT, DECIR_CIENT, LEER_BROMAS,
          DECIR_BROMAS, LEER_NEGACIONES, DECIR_NEGACIONES, NEGATIVAS,
          DESCONOCIDO, CHATEAR, RESP_PREG, Hombres, Mujeres, Nombres, Musica,
          categorias_musica, Videos, categorias_videos, Libros,
          categorias_libros, Wiki, Wikis, name_wikis, categorias_wikis, Artic,
          VJ, categorias_vj, categorias2_vj, Netflix, Netflix_p, categorias_netp
          , Netflix_s, categorias_nets, Type_netflix, Inv, categorias_inv,
          Area_inv
39
40
41         #!git clone https://github.com/NM-Labs/ChatBot.git
42     # path = "D:/GitHub/"
43
44     # path = getDatos()
45
46     Hombres = pd.read_csv(path +'ChatBot/BasesDeDatos/nombreshombres .csv')
47     Mujeres = pd.read_csv(path+'ChatBot/BasesDeDatos/nombresmujeres.csv')
48     Hombres = list(Hombres.iloc[:,0])
49     Mujeres = list(Mujeres.iloc[:,0])
50     Nombres = Hombres + Mujeres
51     Musica = pd.read_csv(path +'ChatBot/BasesDeDatos/Music.csv')
52     Musica = pd.DataFrame(Musica)
53     categorias_musica = list(pd.unique(Musica['terms']))
54     Videos = pd.read_csv(path +'ChatBot/BasesDeDatos/YTVideos.csv')
55     Videos = pd.DataFrame(Videos)
56     categorias_videos = list(pd.unique(Videos['category']))
57     Libros = pd.read_csv(path +'ChatBot/BasesDeDatos/booksdataset.csv')
58     Libros = pd.DataFrame(Libros)
59     categorias_libros = list(pd.unique(Libros['category']))
60     Wiki = pd.read_csv(path +'ChatBot/BasesDeDatos/WIKI.csv')
```

```
61    Wikis = pd.DataFrame(Wiki)
62    name_wikis = list(pd.unique(Wikis['Name']))
63    categorias_wikis = list(pd.unique(Wikis['Vertical1']))
64    Artic = pd.read_csv(path +'ChatBot/BasesDeDatos/ArxivDataClean.csv')
65    Artic = pd.DataFrame(Artic)
66    VJ = pd.read_csv(path +'ChatBot/BasesDeDatos/VGClean.csv')
67    VJ = pd.DataFrame(VJ)
68    categorias_vj = list(pd.unique(VJ['Genre']))
69    categorias2_vj = list(pd.unique(VJ['Platform']))
70    Netflix = pd.read_csv(path +'ChatBot/BasesDeDatos/netflix_titlesClean.csv')
71    Netflix_p = pd.DataFrame(Netflix[Netflix['type']=='Movie'])
72    categorias_netp = list(pd.unique(Netflix_p['listed_in']))
73    Netflix_s = pd.DataFrame(Netflix[Netflix['type']=='TV Show'])
74    categorias_nets = list(pd.unique(Netflix_s['listed_in']))
75    Type_netflix = list(pd.unique(Netflix['type']))
76    Inv = pd.read_csv(path +'ChatBot/BasesDeDatos/InvestigadoresSNIClean.csv')
77    Inv = pd.DataFrame(Inv)
78    categorias_inv = list(pd.unique(Inv['area del Conocimiento']))
79    Area_inv = list(pd.unique(Inv['area del Conocimiento']))
80
81    """# Listas de palabras frases y categorias"""
82
83    OP_ENTRETENIMIENTO = ["videos", "peliculas", "series", "musica", "libros","
          videojuegos","juegos"]
84    OP_ACADEMICO = ["articulo", "investigador", "investigadores", "articulos",
          "definiciones"]
85
86    SALUDOS_IN = ['Hola! Soy MMN Bot, mi especialidad es dar recomendaciones!
          Que tal va tu dia?', 'Hola! Que tal te sientes hoy?', 'Que onda, soy
          MMN Bot! Como te llamas?']
87    SALUDOS = ['hello', 'hi', 'hey', 'hola', 'welcome', 'bonjour', 'greetings',
           'que onda', 'holi']
88    SALUDOS_RESP = ["Hola, es cool hablar contigo!", 'Gusto en conocerte!', "
          Hey - Vamos a platicar un poco!"]
89
90    PREGUNTA_1 = ["Que quisieras que te recomendara, tengo la seccion de
          entretenimiento, academico y covid", "Muy bien, continuemos! Buscas
          algo academico, de entretenimiento o informacion sobre Covid?", "Me
          caes bien, puedo recomendarte algo academico, algo de entretenimiento
          o de Covid, cual prefieres?", "Sos la ostia, tengo para vosotros algo
          de entretenimiento, de covid o algo academico, elige..."]
91
92
93    LEER_NOMBRES = Nombres
```

```
94    DECIR_NOMBRES = ['gusto en conocerte, vamoa platicar :D', 'esta bien curado
          tu nombre, es un gusto.', ", ese nombre mola!, es un gusto conocerte.
      ",'Gusto en conocerte!' , "Hey - Vamos a platicar un poco!"]
95
96    LEER_MUSICA = categorias_musica
97
98
99    LEER_LIBROS = ['no','poco', 'corto', 'medianito', 'menos', 'mucho', '
      bastante', 'largo', 'encanta']
100
101   NOMBRES_LIBROS = categorias_libros
102
103   LEER_VIDEOS = ['entretenimiento', 'peliculas', 'estilo', 'comedia', '
      tecnologia', 'blogs', 'deportes','activismo', 'noticias', 'gaming', '
      educacion', 'animales', 'autos', 'viajes', 'ciencia']
104
105   NOMBRES_VIDEOS = dict(zip(LEER_VIDEOS, categorias_videos))
106   NOMBRES_VIDEOS['ciencia'] = NOMBRES_VIDEOS['tecnologia']
107
108   LEER_INV = ['fisica','matematicas','tierra','biologia','quimica', 'medicina
      ', 'salud', 'humanidades','conducta', 'sociales', 'biotecnologia','
      agropecuarias','ingenierias']
109   DIC_INV = {'fisica': categorias_inv[4], 'matematicas': categorias_inv[4], '
      tierra': categorias_inv[4], 'biologia':categorias_inv[1], 'quimica':
      categorias_inv[1], 'medicina':categorias_inv[3], 'salud':
      categorias_inv[3], 'humanidades':categorias_inv[6], 'conducta':
      categorias_inv[6], 'sociales':categorias_inv[2], 'biotecnologia':
      categorias_inv[0], 'agropecuarias':categorias_inv[0], 'ingenierias':
      categorias_inv[5]}
110
111   LEER_PELIS = ['documentales','accion','comedia','palomera','drama', 'terror
      ', 'clasicos', 'ficcion','infantil']
112   DIC_PELIS = {'documentales': [categorias_netp[x] for x in [0,9,26,27]], '
      accion':[categorias_netp[x] for x in [1,2,16,18,24,32]], 'comedia':[
      categorias_netp[x] for x in [3,4,6,11,13,14,28,31,34]],'palomera':[
      categorias_netp[x] for x in [5]], 'drama':[categorias_netp[x] for x in
       [7,8,15,21,23,25,33]],'terror':[categorias_netp[x] for x in
      [10,19,20]],'clasicos':[categorias_netp[x] for x in [12]],'ficcion':[
      categorias_netp[x] for x in [17]],'infantil':[categorias_netp[x] for x
       in [22,29,30]]}
113
114   LEER_SERIES = ['crimen','novela','infantil','documentales','clasicos', '
      reality']
115   DIC_SERIES = {'crimen': [categorias_nets[x] for x in [0,5,7,9,12,14,19]], '
      novela':[categorias_nets[x] for x in [1,3,4,8,11,16,15]], 'infantil':[
```

```
        categorias_nets[x] for x in [2,18]],'documentales':[categorias_nets[x]
         for x in [6,17]], 'clasicos':[categorias_nets[x] for x in [10,13]],'
        reality':[categorias_nets[x] for x in [20,21]]}
116
117     LEER_VJ_P = ['xbox','360','one','play','station','playstation', 'wii', 'psp
        ', 'computadora', 'compu', 'pc']
118     DIC_VJ_P = {'xbox': [categorias2_vj[x] for x in [4,13,17]], '360': [
        categorias2_vj[x] for x in [4,13,17]],'one': [categorias2_vj[x] for x
        in [4,13,17]], 'playstation':[categorias2_vj[x] for x in
        [5,6,10,12,16]],'play':[categorias2_vj[x] for x in [5,6,10,12,16]],'
        station':[categorias2_vj[x] for x in [5,6,10,12,16]],'psp':[
        categorias2_vj[x] for x in [5,6,10,12,16]], 'wii':[categorias2_vj[x]
        for x in [0,19]],'computadora':[categorias2_vj[x] for x in [14]],'
        compu':[categorias2_vj[x] for x in [14]],'pc':[categorias2_vj[x] for x
         in [14]]}
119     LEER_VJ_G = ['deportes','plataforma','carreras','roles','rompecabezas','
        variado', 'disparos', 'simulacion', 'accion', 'peleas', 'aventura', "
        estrategia"]
120     DIC_VJ_G = dict(zip(LEER_VJ_G, categorias_vj))
121
122     LEER_CATEGORIAS = ['libros', 'libro', 'musica', 'videos', 'video', 'si', '
        leer']
123     DIC_LIBROS = {'no': 'short', 'poco': 'short', 'corto': 'short','medianito':
         'medium','mediano': 'medium','menos': 'medium', 'mucho': 'large', '
        bastante': 'large','encanta': 'large' ,'largo': 'large' }
124     LEER_COVID = ['cuarentena', 'covid', 'coronavirus', 'encerramiento', 'd 19'
        , 'sars', 'corona']
125
126
127     LEER_COMPU = ['python', 'codigo', 'computadora', 'algoritmo', ]
128     DECIR_COMPU = ["Python es de lo que estoy hecho.", \
129                "Sabias que estoy hecho con codigo!?", \
130                "Las computadoras son magicas", \
131                "Crees que podria pasar el Test de Turing?"]
132
133     LEER_CIENT = ['turing', 'hopper', 'neumann', 'lovelace']
134     DECIR_CIENT = ['fue asombroso!', 'hizo muchas cosas importantes!', 'es
        alguien del que que valdria la pena saber mas :).']
135     NOMBRES_CIENT = {'turing': 'Alan', 'hopper': 'Grace', 'neumann': 'John von'
        , 'lovelace': 'Ada'}
136
137     LEER_BROMAS = ['divertido', 'gracioso', 'ja', 'jaja', 'jajaja', 'xD']
138     DECIR_BROMAS = ['ja!', 'jajaja!', 'XD', 'lol']
139
140     LEER_NEGACIONES = ['matlab', 'java', 'C++']
```

```
141     DECIR_NEGACIONES = ["No, lo siento. :(, No me gustaria hablar por ahora de
            eso."]
142
143     NEGATIVAS = ['no', "no no", 'nop', 'nunca', "negativo", "ninguno"]
144
145     DESCONOCIDO = ['Bien.', 'Okay', 'Mm?', 'Si!', 'bien...', 'Nam', 'Hum']
146     CHATEAR = ['Que te gustaria hacer ahora?, puedo recomedarte algo de musica,
             libros o algun video enretenido, Cual te gustaria?', 'Veamos, Que
            tipo de musica te gusta?', 'Quieres algo para relajarte?', 'Puedo
            buscar algo de buena musica para ti,Que genero te gusta?', 'Tengo
            algunos videos entretenidos!, escoge una categoria :D','Te gustan los
            videos? Tengo de diferentes categorias', 'ademas, tengo aqui algunos
            de mis libros favoritos, te gusta leer mucho, mas o menos, o solo un
            poco?',
147                 'Sobre que deberia buscar?']
148
149     RESP_PREG = "Soy demasiado timido para a responder eso, jeje. De que otra
            cosa te gustaria una recomendacion?"
150
151 # return
152 ### Funciones ###
153
154 def es_pregunta(entrada):
155   for i in entrada:
156     if i == '?':
157       salida = True
158     else:
159       salida = False
160   return salida
161
162 def quitar_acentos(string):
163     acentos = set(map(unicodedata.lookup, ('COMBINING ACUTE ACCENT', 'COMBINING
             GRAVE ACCENT', 'COMBINING TILDE')))
164     chars = [c for c in unicodedata.normalize('NFD', string) if c not in
            acentos]
165     return unicodedata.normalize('NFC', ''.join(chars))
166
167 def remover_puntuacion(entrada):
168   out_string = ""
169   for i in entrada:
170     if i not in string.punctuation:
171       out_string += i
172   return out_string
173
174 def preparar_texto(entrada):
```

```
175   temp_string = entrada.lower()
176   temp_string = remover_puntuacion(temp_string)
177   temp_string = quitar_acentos(temp_string)
178   lista_salida = temp_string.split()
179   return lista_salida
180
181 def responder_echo(entrada, numero_bromas,espaciador):
182   if entrada != None:
183     echo_salida = (entrada + espaciador) * numero_bromas
184   else:
185     echo_salida = None
186   return echo_salida
187
188 def selector(lista_entrada, checar_lista, regresar_lista):
189   salida = None
190   for i in lista_entrada:
191     if i in checar_lista:
192       salida = random.choice(regresar_lista)
193       break
194   return salida
195
196 def concatenar_string(string1, string2, separador):
197   salida = string1 + separador + string2
198   return salida
199
200 def lista_a_cadena(lista_entrada, separador):
201   salida = lista_entrada[0]
202   for i in lista_entrada[1:]:
203     salida = concatenar_string(salida, i, separador)
204   return salida
205
206 def esta_en_lista(lista_uno, lista_dos): #Checar si cualqueir elemento esta en
      dos listas.
207
208     for elemento in lista_uno:
209         if elemento in lista_dos:
210             return True
211     return False
212
213 def encontrar_en_lista(lista_uno, lista_dos): # Find and return an element
     from list_one that is in list_two, or None otherwise.
214     for elemento in lista_uno:
215         if elemento in lista_dos:
216             return elemento
217     return None
```

```
218
219 def terminar_chat(lista_entrada):
220   if encontrar_en_lista(lista_entrada, ["no", "adios","nelson","bye","chao","
        vemos","nel"]):
221     salida = True
222   else:
223     salida = False
224   return salida
225
226
227 def contar_puntos(entrada):
228   p = 0
229   h = []
230   for i in entrada:
231     o = i.count('.')
232     if o == 1:
233       p += 1
234       if p == 5:
235         break
236     h.append(i)
237   h.append('.')
238   return h
239
240 def codigo_pais(nombre):
241     try:
242         return pycountry.countries.lookup(nombre).alpha_3
243     except:
244         return None
245
246 def creargrafica():
247     wget.download("https://raw.githubusercontent.com/CSSEGISandData/COVID-19/
          master/csse_covid_19_data/csse_covid_19_time_series/
          time_series_covid19_confirmed_global.csv", bar=None)
248     df_confirm = pd.read_csv('time_series_covid19_confirmed_global.csv')
249     df_confirm = df_confirm.drop(columns=['Province/State','Lat', 'Long'])
250     df_confirm = df_confirm.groupby('Country/Region').agg('sum')
251     date_list = list(df_confirm.columns)
252     df_confirm['country'] = df_confirm.index
253     df_confirm['iso_alpha_3'] = df_confirm['country'].apply(codigo_pais)
254     df_long = pd.melt(df_confirm, id_vars=['country','iso_alpha_3'], value_vars
          =date_list)
255     fig = px.choropleth(df_long, # Input Dataframe
256                         locations="iso_alpha_3", # identify country code column
257                         color="value", # identify representing column
258                         hover_name="country", # identify hover name
```

```
259                     animation_frame="variable", # identify date column
260                     projection="natural earth", # select projection
261                     color_continuous_scale = 'Turbo', # select prefer color
                            scale
262                     range_color=[0,50000] # select range of dataset
263                     )
264     os.remove("time_series_covid19_confirmed_global.csv")
265     return fig.show()
266
267 def escuchar_mensaje(tunombre="INPUT", w=False):
268     texto_salida_audio = None
269     mensaje = None
270     with SRG.Microphone() as s:
271         print(chr(27)+"[1;31m"+'Estoy escuchando...')
272         entrada_audio = st.record(s, duration=5)
273         # sys.stdout.write("\033[F")
274         try:
275             print(chr(27)+"[1;31m"+"Procesando...")
276             texto_salida_audio = st.recognize_google(entrada_audio,language="es"
                    )
277             # print(texto_salida_audio)
278             print(chr(27)+"[1;31m"+"Reconocido.")
279             print(chr(27)+"[1;30m"+str(tunombre) +': \t' + str(
                    texto_salida_audio))
280             if w:
281                 mensaje = texto_salida_audio.split() #w se usa para Wikipedia
282                 mensaje = [mensaje.capitalize() for mensaje in mensaje]
283             else:
284                 mensaje = preparar_texto(texto_salida_audio)
285         except:
286             print(chr(27)+"[1;31m"+"No he podido escucharte, intenta de nuevo")
287
288             mensaje = escuchar_mensaje(tunombre)
289     return mensaje
290
291 def hablar(msg_salida):
292     sentencia = pyttsx3.init()
293
294     sentencia.setProperty("rate",150)
295     sentencia.setProperty("volume",.6)
296     listVoices = sentencia.getProperty("voices")
297     sentencia.setProperty("voice",listVoices[0].id)
298
299     sentencia.say(msg_salida)
300     sentencia.runAndWait()
```

```
301
302 ### Caso 2 ###
303 def videos(tunombre):
304     msg_salida = None
305     msg_salida = random.choice(["Que tipos de videos te gustarian?, tengo de:\n
            ", "Genial!, tengo estas categorias:\n", "Muy bien, revisare mi
            coleccion favorita de videos, podriamos empezar por: \n"])
306     opciones_videos = " - Comedia\n - Tecnologia\n - Peliculas\n - Estilo\n -
            Entretenimiento\n - Blogs\n - Deportes\n - Activismo\n - Noticias\n -
            Gaiming\n - Educacion\n - Animales\n - Autos\n - Viajes\n - Ciencia"
307     # msg_salida = msg_salida + opciones_videos
308     print(chr(27)+"[1;34m"+'CHATBOT:')
309     hablar(msg_salida+opciones_videos)
310     for i in textwrap.wrap(str(msg_salida), 130):
311       print(chr(27)+"[1;34m"+ i)
312     print("\n"+chr(27)+"[1;34m"+'CHATBOT:\n'+opciones_videos)
313     msg = escuchar_mensaje(tunombre)
314     name = encontrar_en_lista(list(msg), LEER_VIDEOS)
315     if name:
316         ran = np.random.randint(0,len(Videos[Videos['category']==NOMBRES_VIDEOS
                [name]]))
317         title = Videos[Videos['category']==NOMBRES_VIDEOS[name]][['title', '
                video_id']]
318         msg_salida = (
319             'Si te gusta {} yo te recomendaria este video "{}" , lo puedes ver
                    en https://www.youtube.com/watch?v={}'.format(name,
320                 title.iloc[ran][0], title.iloc[ran][1]))
321     else:
322         msg_salida = "Hum... creo que no capte algo de lo que dijiste, podrias
                repetirlo?"
323         print(chr(27)+"[1;34m"+'CHATBOT:')
324         hablar(msg_salida)
325         for i in textwrap.wrap(str(msg_salida), 130):
326           print(chr(27)+"[1;34m"+ i)
327         videos(tunombre)
328         msg_salida = []
329
330     print(chr(27)+"[1;34m"+'CHATBOT:')
331     hablar(msg_salida)
332     for i in textwrap.wrap(str(msg_salida), 130):
333       print(chr(27)+"[1;34m"+ i)
334
335
336     return
337
```

```
338 def peliculas(tunombre):
339     msg_salida = None
340     msg_salida = random.choice(["Que clase de peliculas te gustan?, tengo\n", "
            Genial!, tengo estas categorias\n", "Muy bien, entretenimiento,
            podriamos empezar por: \n"])
341     opciones_peliculas = " - documentales\n - accion\n - comedia\n - palomera\n
            - drama\n - terror\n - clasicos\n - ficcion\n - infantil"
342     # msg_salida = msg_salida + opciones_peliculas
343     print(chr(27)+"[1;34m"+'CHATBOT:')
344     hablar(msg_salida+opciones_peliculas)
345     for i in textwrap.wrap(str(msg_salida), 130):
346       print(chr(27)+"[1;34m"+ i)
347     print("\n"+chr(27)+"[1;34m"+'CHATBOT:\n'+opciones_peliculas)
348     msg = escuchar_mensaje(tunombre)
349     name = encontrar_en_lista(msg, LEER_PELIS)
350     if name:
351         select = random.choice(DIC_PELIS[name])
352         ran = np.random.randint(0,len(Netflix_p[Netflix_p['listed_in']==select
                ]))
353         datos = Netflix_p[Netflix_p['listed_in']==select][['title', 'duration',
                'description']].iloc[ran]
354         msg_salida = (
355             'Si te gusta la categoria de {} yo te recomendaria la pelicula "{}",
                    que dura {}, trata de: {}'.format(name,
356                 datos[0], GoogleTranslator(source='auto', target='es').translate
                        (datos[1]), GoogleTranslator(source='auto', target='es').
                        translate(datos[2])))
357     else:
358         msg_salida = "Hum... creo que no capte algo de lo que dijiste, podrias
                repetirlo?"
359         print(chr(27)+"[1;34m"+'CHATBOT:')
360         hablar(msg_salida)
361         for i in textwrap.wrap(str(msg_salida), 130):
362           print(chr(27)+"[1;34m"+ i)
363         peliculas(tunombre)
364         msg_salida = []
365 
366     print(chr(27)+"[1;34m"+'CHATBOT:')
367     hablar(msg_salida)
368     for i in textwrap.wrap(str(msg_salida), 130):
369       print(chr(27)+"[1;34m"+ i)
370 
371 
372     return
373 
```

```
374 def series(tunombre):
375     msg_salida = None
376     msg_salida = random.choice(["Que tipo de series te gustan?, tengo\n", "
            Genial!, tengo estas categorias\n", "Muy bien, series, podriamos
            empezar por: \n"])
377     categorias_series = " - Crimen\n - Novela\n - Infantil\n - Documentales\n -
             Clasicos\n - Reality Shows"
378     # msg_salida = msg_salida + categorias_entretenimiento
379     print(chr(27)+"[1;34m"+'CHATBOT:')
380     hablar(msg_salida+" - Crimen - Novela - Infantil - Documentales - Clasicos
            - Reality Schows")
381     for i in textwrap.wrap(str(msg_salida), 130):
382       print(chr(27)+"[1;34m"+ i)
383     print("\n"+chr(27)+"[1;34m"+'CHATBOT:\n'+categorias_series)
384     msg = escuchar_mensaje(tunombre)
385     name = encontrar_en_lista(msg, LEER_SERIES)
386     if name:
387         select = random.choice(DIC_SERIES[name])
388         ran = np.random.randint(0,len(Netflix_s[Netflix_s['listed_in']==select
                ]))
389         datos = Netflix_s[Netflix_s['listed_in']==select][['title', 'duration',
                'description']].iloc[ran]
390         msg_salida = (
391     'Si te gusta la categoria de {} yo te recomendaria la serie "{}", que tiene
             {}, trata de: {}'.format(name,
392         datos[0], GoogleTranslator(source='auto', target='es').translate(datos
                [1]), GoogleTranslator(source='auto', target='es').translate(datos
                [2])))
393     else:
394         msg_salida = "Hum... creo que no capte algo de lo que dijiste, podrias
                repetirlo?"
395         print(chr(27)+"[1;34m"+'CHATBOT:')
396         hablar(msg_salida)
397         for i in textwrap.wrap(str(msg_salida), 130):
398           print(chr(27)+"[1;34m"+ i)
399         series(tunombre)
400         msg_salida = []
401
402     print(chr(27)+"[1;34m"+'CHATBOT:')
403     hablar(msg_salida)
404     for i in textwrap.wrap(str(msg_salida), 130):
405       print(chr(27)+"[1;34m"+ i)
406
407
408     return
```

```
409
410 def musica(tunombre):
411     msg_salida = None
412     msg_salida = random.choice(["Que genero de musica prefieres?\n", "Genial!,
            Que musica te gusta?\n", "La musica es genial!, podriamos empezar por
            decirme tu genero favorito \n"])
413
414     print(chr(27)+"[1;34m"+'CHATBOT:')
415     hablar(msg_salida)
416     for i in textwrap.wrap(str(msg_salida), 130):
417       print(chr(27)+"[1;34m"+ i)
418
419     msg = escuchar_mensaje(tunombre)
420     name = encontrar_en_lista(msg, LEER_MUSICA)
421     if name:
422         ran = np.random.randint(0,len(Musica[Musica['terms']==name]))
423         title = Musica[Musica['terms']==name][['release.name', 'artist.name']]
424         msg_salida = (
425             'Si te gusta el {} te recomiendo esta cancion "{}" de {}'.format(
                    name,
426                 title.iloc[ran][0], title.iloc[ran][1]))
427     else:
428         msg_salida = "Hum... creo que no capte algo de lo que dijiste, podrias
                repetirlo?"
429         print(chr(27)+"[1;34m"+'CHATBOT:')
430         hablar(msg_salida)
431         for i in textwrap.wrap(str(msg_salida), 130):
432           print(chr(27)+"[1;34m"+ i)
433         musica(tunombre)
434         msg_salida = []
435
436     print(chr(27)+"[1;34m"+'CHATBOT:')
437     hablar(msg_salida)
438     for i in textwrap.wrap(str(msg_salida), 130):
439       print(chr(27)+"[1;34m"+ i)
440
441
442     return
443
444 def libros(tunombre):
445     msg_salida = None
446     msg_salida = random.choice(["Que tanto te gusta leer?, mucho, poco?\n", "
            Genial!, podria sugerirte un libro corto, medianito o algo largo.\n",
            "Muy bien, libros, que tan grandes? mucho, mas o menos, poco...: \n"])
447
```

```
448     print(chr(27)+"[1;34m"+'CHATBOT:')
449     hablar(msg_salida)
450     for i in textwrap.wrap(str(msg_salida), 130):
451       print(chr(27)+"[1;34m"+ i)
452 
453     msg = escuchar_mensaje(tunombre)
454     name = encontrar_en_lista(msg, LEER_LIBROS)
455     if name:
456         ran = np.random.randint(0,len(Libros[Libros['category']==DIC_LIBROS[
                name]]))
457         title = Libros[Libros['category']==DIC_LIBROS[name]][['title', 'authors
                ', 'num_pages']]
458         msg_salida = (
459          'Este libro "{}" suena bien para ti, fue escrito por {} y tiene {}
                paginas.'.format(
460             title.iloc[ran][0], title.iloc[ran][1], title.iloc[ran][2]))
461     else:
462         msg_salida = "Hum... creo que no capte algo de lo que dijiste, podrias
                repetirlo?"
463         print(chr(27)+"[1;34m"+'CHATBOT:')
464         hablar(msg_salida)
465         for i in textwrap.wrap(str(msg_salida), 130):
466           print(chr(27)+"[1;34m"+ i)
467         libros(tunombre)
468         msg_salida = []
469 
470     print(chr(27)+"[1;34m"+'CHATBOT:')
471     hablar(msg_salida)
472     for i in textwrap.wrap(str(msg_salida), 130):
473       print(chr(27)+"[1;34m"+ i)
474 
475 
476     return
477 
478 def juegos(tunombre):
479     msg_salida = None
480     msg_salida = random.choice(["Que clase de videojuegos te gustan?, que
            plataforma usas?\n", "Genial!, dime una categoria y plataforma.\n", "
            Muy bien, videojuegos, de que tipo, que consola?: \n"])
481     categorias_videojuegos = " Categorias:\t Consolas:\n - Accion\t + Xbox\n -
            Aventuras\t + PlayStation\n - Carreras\t + Wii\n - Deportes\t +
            Computadora\n - Disparos\n - Estrategia\n - Peleas\n - Plataforma\n -
            Roles\n - Rompecabezas\n - Simulacion\n - Variado"
482     # msg_salida = msg_salida + categorias_videojuegos
483     print(chr(27)+"[1;34m"+'CHATBOT:')
```

```
484     hablar(msg_salida+" Las consolas disponibles son: Xbox PlayeStation Wii y
            Computadora\n con las siguientes categorias")
485     for i in textwrap.wrap(str(msg_salida), 130):
486       print(chr(27)+"[1;34m"+ i)
487     print("\n"+chr(27)+"[1;34m"+'CHATBOT:\n'+categorias_videojuegos)
488     msg = escuchar_mensaje(tunombre)
489     keyp = encontrar_en_lista(msg, LEER_VJ_P)
490     keyg = encontrar_en_lista(msg, LEER_VJ_G)
491     if keyp and keyg:
492         Keyp = random.choice(DIC_VJ_P[keyp])
493         Keyg = DIC_VJ_G[keyg]
494         VJG = VJ[VJ['Genre'] == Keyg]
495         VJG = VJG[VJG['Platform'] == Keyp]
496         ran = np.random.randint(0,len(VJG))
497         datos = VJG[['Name','Genre', 'Platform']].iloc[ran]
498         msg_salida = (
499             'Si te gusta la categoria de {} yo te recomendaria "{}", para {}.'.
                    format(keyg,
500                 datos[0], datos[2]))
501     else:
502         msg_salida = "Hum... creo que no capte algo de lo que dijiste, podrias
                repetirlo?"
503         print(chr(27)+"[1;34m"+'CHATBOT:')
504         hablar(msg_salida)
505         for i in textwrap.wrap(str(msg_salida), 130):
506           print(chr(27)+"[1;34m"+ i)
507         juegos(tunombre)
508         msg_salida = []
509
510     print(chr(27)+"[1;34m"+'CHATBOT:')
511     hablar(msg_salida)
512     for i in textwrap.wrap(str(msg_salida), 130):
513       print(chr(27)+"[1;34m"+ i)
514
515
516     return
517
518 def articulos(tunombre):
519     msg_salida = None
520     msg_salida = random.choice(["Me agrada que quieras descubrir conocimiento,
            di una palabra clave (en ingles)\n", "Genial! dime una palabra clave (
            en ingles), para encontrar uno interesante\n", "Muy bien, busquemos
            uno interesante, dime una palabra clave que podria interesarte: \n"])
521     print(chr(27)+"[1;34m"+'CHATBOT:')
522     hablar(msg_salida)
```

```
523     for i in textwrap.wrap(str(msg_salida), 130):
524       print(chr(27)+"[1;34m"+ i)
525 
526     subartic = []
527 
528     w = escuchar_mensaje(tunombre,w=True)
529     print(chr(27)+"[1;34m"+'Buscando alguna coincidencia...')
530     w = random.choice(w)
531     try:
532         subArtic = Artic[Artic['title'].str.contains(w)]
533         # print(subArtic)
534     # if subartic != None:
535         ran = np.random.randint(0,len(subArtic))
536         title = Artic[['title']].iloc[ran][0]
537         id = Artic[['id']].iloc[ran][0]
538         msg_salida = (
539             'Un articulo relacionado a {} que encontre para ti: {}'.format(w,
                    GoogleTranslator(source='auto', target='es').translate(title))
                    +
540              ' Puedes leerlo completo en: https://arxiv.org/abs/{}'.format(id))
541     except:
542         msg_salida = "Hum... creo que no capte algo de lo que dijiste, podrias
                repetirlo?"
543         print(chr(27)+"[1;34m"+'CHATBOT:')
544         hablar(msg_salida)
545         for i in textwrap.wrap(str(msg_salida), 130):
546           print(chr(27)+"[1;34m"+ i)
547         articulos(tunombre)
548         msg_salida = []
549 
550     print(chr(27)+"[1;34m"+'CHATBOT:')
551     hablar(msg_salida)
552     for i in textwrap.wrap(str(msg_salida), 130):
553       print(chr(27)+"[1;34m"+ i)
554 
555     return
556 
557 def wikis(tunombre):
558     msg_salida = None
559     msg_salida = random.choice(["Que te gustaria saber?, preguntame algun
            concepto\n", "Genial!, te interesa saber la definicion de algo en
            particular?\n", "Muy bien, podriamos empezar por algo que quieras
            saber... \n"])
560     print(chr(27)+"[1;34m"+'CHATBOT:')
561     hablar(msg_salida)
```

```
562     for i in textwrap.wrap(str(msg_salida), 130):
563       print(chr(27)+"[1;34m"+ i)
564 
565     w = escuchar_mensaje(tunombre,w=True)
566     w = GoogleTranslator(source='auto', target='en').translate(lista_a_cadena(w
            , ''))
567     w = w.split()
568     name = encontrar_en_lista(w, name_wikis)
569     if name:
570         ran = np.random.randint(0,len(Wikis[Wikis['Name']==name]))
571         title = Wikis[Wikis['Name']==name][['Name', 'WikiDescription','WikiUrl'
                ]]
572         msg_salida = (
573             'Aqui esta la definicion de {} que encontre para ti: {}'.format(
574                 GoogleTranslator(source='auto', target='es').translate(
                        title.iloc[ran][0]), GoogleTranslator(source='auto', target=
                        'es').translate(lista_a_cadena(contar_puntos(title.iloc[ran
                        ][1]), ''))) +
575              ' Puedes leer mas de ello en: {}'.format(
576                 title.iloc[ran][2]))
577     else:
578         msg_salida = "Hum... creo que no capte algo de lo que dijiste, podrias
                repetirlo?"
579         print(chr(27)+"[1;34m"+'CHATBOT:')
580         hablar(msg_salida)
581         for i in textwrap.wrap(str(msg_salida), 130):
582           print(chr(27)+"[1;34m"+ i)
583         wikis(tunombre)
584         msg_salida = []
585 
586     print(chr(27)+"[1;34m"+'CHATBOT:')
587     hablar(msg_salida)
588     for i in textwrap.wrap(str(msg_salida), 130):
589       print(chr(27)+"[1;34m"+ i)
590 
591     return
592 
593 def investigadores(tunombre):
594     msg_salida = None
595     msg_salida = random.choice(["Que area del conocimiento te agrada en este
            momento?, estas son:\n", "Genial!, las areas en las que podria
            encontrar a alguien son\n", "Muy bien, en que area estas interesado: \
            n"])
596     categorias_inv = " - Fisica, Matematicas y Ciencias de la Tierra\n -
            Biologia y Quimica\n - Medicina y Ciencias de la Salud\n - Humanidades
```

```
        y Ciencias de la Conducta\n - Ciencias Sociales\n - Biotecnologia y
        Ciencias Agropecuarias\n - Ingenierias"
597     # msg_salida = msg_salida + categorias_inv
598     print(chr(27)+"[1;34m"+'CHATBOT:')
599     hablar(msg_salida+categorias_inv)
600     for i in textwrap.wrap(str(msg_salida), 130):
601       print(chr(27)+"[1;34m"+ i)
602     print("\n"+chr(27)+"[1;34m"+'CHATBOT:\n'+categorias_inv)
603     msg = escuchar_mensaje(tunombre)
604     name = encontrar_en_lista(msg, LEER_INV)
605     if name:
606         ran = np.random.randint(0,len(Inv[Inv['area del Conocimiento']==DIC_INV
            [name]]))
607         datos = Inv[Inv['area del Conocimiento']==DIC_INV[name]][['Nombre
            Completo', 'area del Conocimiento', 'Institucion de Adscripcion']].
            iloc[ran]
608         msg_salida = (
609             'Si te gusta el area de {} yo te recomendaria contactar o buscar el
                trabajo desarrollado por "{}", adscrito a {}'.format(datos[1],
                datos[0], datos[2]))
610     else:
611         msg_salida = "Hum... creo que no capte algo de lo que dijiste, podrias
            repetirlo?"
612         print(chr(27)+"[1;34m"+'CHATBOT:')
613         hablar(msg_salida)
614         for i in textwrap.wrap(str(msg_salida), 130):
615           print(chr(27)+"[1;34m"+ i)
616         investigadores(tunombre)
617         msg_salida = []
618
619     print(chr(27)+"[1;34m"+'CHATBOT:')
620     hablar(msg_salida)
621     for i in textwrap.wrap(str(msg_salida), 130):
622       print(chr(27)+"[1;34m"+ i)
623
624
625     return
626
627 def covids(tunombre):
628     msg_salida = None
629     msg_salida = random.choice(["Muy bien! Aqui hay algo de informacion sobre
            Covid. Cuida tu salud!\n Te dejo los datos actualizados sobre covid en
             Mexico\n Y un mapa interactivo de la evolucion de covid en el mundo."
            ])
630     print(chr(27)+"[1;34m"+'CHATBOT:')
```

```
631     hablar(msg_salida)
632     for i in textwrap.wrap(str(msg_salida), 130):
633       print(chr(27)+"[1;34m"+ i)
634     covid = Covid()
635     casos = covid.get_status_by_country_name(country_name='mexico')
636     print(chr(27)+"[1;31m"+'CHATBOT: Aqui hay un poco de informacion
            actualizada de Covid en Mexico: \n')
637     for x in casos:
638         print(chr(27)+"[1;31m"+ x, ':', casos[x])
639     creargrafica()
640 
641     return
642 
643 ### Switchs ###
644 
645 def switcher_entretenimiento(key,tunombre):
646     switch_entretenimiento = {
647         "videos": videos,
648         "peliculas": peliculas,
649         "series": series,
650         "musica": musica,
651         "libros": libros,
652         "videojuegos": juegos,
653         "juegos": juegos
654     }
655     funcion = switch_entretenimiento.get(key)
656     return funcion(tunombre)
657 
658 def switcher_academico(key,tunombre):
659     switch_academico = {
660         "articulos": articulos,
661         "articulo": articulos,
662         "investigadores": investigadores,
663         "investigador": investigadores,
664         "definicion": wikis,
665         "definiciones": wikis,
666     }
667     funcion = switch_academico.get(key)
668     return funcion(tunombre)
669 
670 
671 ### Casos 1###
672 def entretenimiento(tunombre):
673     msg_salida = None
```

```
674     msg_salida = random.choice(["Que te gustaria de entretenimiento?, tengo\n",
            "Genial!, tengo estas categorias\n", "Muy bien, entretenimiento,
            podriamos empezar por: \n"])
675     categorias_entretenimiento = " - Videos\n - Peliculas\n - Series\n - Musica
            \n - Libros\n - Videojuegos"
676     # msg_salida = msg_salida + categorias_entretenimiento
677     print(chr(27)+"[1;34m"+'CHATBOT:')
678     hablar(msg_salida+categorias_entretenimiento)
679     for i in textwrap.wrap(str(msg_salida), 130):
680       print(chr(27)+"[1;34m"+ i)
681     print("\n"+chr(27)+"[1;34m"+'CHATBOT:\n'+categorias_entretenimiento)
682     msg = escuchar_mensaje(tunombre)
683     key = encontrar_en_lista(msg, OP_ENTRETENIMIENTO)
684     if key:
685         switcher_entretenimiento(key,tunombre)
686     else:
687         msg_salida = "Hum... creo que no capte algo de lo que dijiste, podrias
                repetirlo?"
688         print(chr(27)+"[1;34m"+'CHATBOT:')
689         hablar(msg_salida)
690         for i in textwrap.wrap(str(msg_salida), 130):
691           print(chr(27)+"[1;34m"+ i)
692         entretenimiento(tunombre)
693         msg_salida = []
694     return
695
696 def academico(tunombre):
697     msg_salida = None
698     msg_salida = random.choice(["Tengo distintas recomendaciones academicas,
            algunas son:\n", "Genial!, tengo estas categorias\n", "Muy bien, el
            ambito academico, podriamos empezar por: \n"])
699     categorias_academico = " - Articulos\n - Investigadores\n - Definiciones"
700     # msg_salida = msg_salida + categorias_academico
701     print(chr(27)+"[1;34m"+'CHATBOT:')
702     hablar(msg_salida+categorias_academico)
703     for i in textwrap.wrap(str(msg_salida), 130):
704       print(chr(27)+"[1;34m"+ i)
705     print("\n"+chr(27)+"[1;34m"+'CHATBOT:\n'+categorias_academico)
706     msg = escuchar_mensaje(tunombre)
707     key = encontrar_en_lista(msg, OP_ACADEMICO)
708     if key:
709         switcher_academico(key,tunombre)
710     else:
711         msg_salida = "Hum... creo que no capte algo de lo que dijiste, podrias
                repetirlo?"
```

```
712         print(chr(27)+"[1;34m"+'CHATBOT:')
713         hablar(msg_salida)
714         for i in textwrap.wrap(str(msg_salida), 130):
715           print(chr(27)+"[1;34m"+ i)
716         academico(tunombre)
717         msg_salida = []
718 
719     return
720 
721 def switcher_general(key,tunombre):
722     switch_general = {
723         "entretenimiento": entretenimiento,
724         "academico": academico,
725         "covid": covids
726     }
727 
728     funcion = switch_general.get(key)
729     return funcion(tunombre)
730 
731 def general(tunombre):
732     msg = escuchar_mensaje(tunombre)
733     primer_mensaje = encontrar_en_lista(msg, ["entretenimiento","academico","
            covid"])
734     if primer_mensaje:
735         switcher_general(primer_mensaje, tunombre)
736     else:
737         msg_salida = "Hum... creo que no capte algo de lo que dijiste, podrias
                repetirlo?"
738         print(chr(27)+"[1;34m"+'CHATBOT:')
739         hablar(msg_salida)
740         for i in textwrap.wrap(str(msg_salida), 130):
741           print(chr(27)+"[1;34m"+ i)
742         general(tunombre)
743         msg_salida = []
744     return
745 
746 def chatear(path):
747     """funcion principal para tener un chat."""
748     getDatos(path)
749     print(chr(27)+"[1;34m"+'Que tal! Soy tu amigo MMN Bot! Cual es tu nombre?:
            \n')
750     hablar('Que tal! Soy tu amigo MMN Bot! Cual es tu nombre?:')
751     chat = True
752     tunombre = None
753     with SRG.Microphone() as s:
```

```
754             print('Estoy escuchando...')
755             entrada_audio = st.record(s, duration=5)
756             sys.stdout.write("\033[F")
757             try:
758                 texto_salida = st.recognize_google(entrada_audio,language="es")
759             except:
760                 print("No he podido escucharte, intenta de nuevo")
761                 texto_salida = escuchar_mensaje(tunombre)
762     if tunombre != None:
763       msg = texto_salida#input(chr(27)+"[1;30m"+str(tunombre) +': \t') \\ chr
              (27)+"[1;30m"+str(tunombre) +': \t' + texto_salida
764       print(chr(27)+"[1;30m"+str(tunombre) +': \t' + msg)
765     else:
766       msg = texto_salida#input(chr(27)+"[1;30m"+'INPUT : \t')
767       print(chr(27)+"[1;30m"+'INPUT : \t' + msg)
768       n = msg.upper() # n sirve para la funcion de nombres
769       n = n.split() # en lugar de la funcion preparar_texto
770       for i in n:
771         i = [i]
772         if esta_en_lista(i, LEER_NOMBRES):
773             tunombre = encontrar_en_lista(i, LEER_NOMBRES)
774             msg_salida =(lista_a_cadena([tunombre.capitalize(),
775                                          selector(i, LEER_NOMBRES, DECIR_NOMBRES)],
                                               ' '))
776     while chat:
777         msg = None
778         msg_salida = None
779         preg1 = random.choice(PREGUNTA_1)
780         print(chr(27)+"[1;34m"+'CHATBOT: \t'+ preg1)
781         hablar(preg1)
782
783         general(tunombre)
784         msg_salida = "Genial, un gusto hablar contigo quieres continuar
                conversando?"
785         print(chr(27)+"[1;34m"+'CHATBOT: \t'+ msg_salida)
786         hablar(msg_salida)
787         msg = escuchar_mensaje(tunombre)
788         if terminar_chat(msg):
789             msg_salida = 'Adios!'
790             print(chr(27)+"[1;34m"+'CHATBOT: \t'+ msg_salida)
791             hablar(msg_salida)
792             chat = False
```

- **Python Jupyter Notebook ChatbotSR.ipynb**

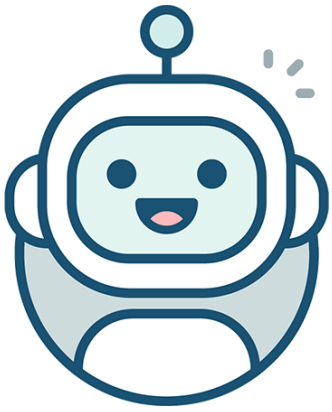
Librerías

Corre esta celda

In [3]:
```
## Solo correr la primera vez que lo utilizas, despues ya no.
!pip install covid
!pip install pycountry
!pip install plotly
!pip install wget
!pip install deep-translator
!pip install nltk
!pip install pandas
!pip install numpy
!pip install opencv-python
!pip install unicodedata2
!pip install pyaudio
!pip install pyttsx3
# !pip install os-win
# !pip install string
# !pip install random
# !pip install os-sys
```

Interactuar

Corre la celda de abajo para interactuar con MMNBot. Puedes interactuar con el de forma normal, solo intenta seguir la corriente.



In [2]:

```
#@title MMN Chatbot
#@markdown ![Esta es una imagen de ejemplo] \\
    (https://encrypted-tbn0.gstatic.com/images?q=tbn \\
    3AANd9GcRGbLPwoNkbvHyZMHYn5F-8gnr2rF7BPKXnFw&usqp=CAU)
import chatVR as c
c.chatear("Pon tu ruta a la carpeta del ChatBot aqu")
```

Referencias

Mr. Brijain, R Patel, Mr. Kushik, and K Rana. A survey on decision tree algorithm for classification.

Menal Dahiya. A tool of conversation: Chatbot. *International Journal of Computer Sciences and Engineering*, 5 (5):158–161, 2017.

Davuluri Hemanth. Decision Trees Explained With a Practical Example – Towards AI — The Best of Tech, Science, and Engineering, may 2020. URL https://towardsai.net/p/programming/decision-trees-explained-with-a-practical-example-fe47872d3b53.

J. R. Quinlan. Learning decision tree classifiers. *ACM Comput. Surv.*, 28(1):71–72, March 1996. ISSN 0360-0300. doi: 10.1145/234313.234346.

A. Shah, B. Jain, B. Agrawal, S. Jain, and S. Shim. Problem solving chatbot for data structures. In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 184–189, 2018. doi: 10.1109/CCWC.2018.8301734.