Chatbot para soporte a usuarios

David Eduardo Gallardo Fernández 0931556
Monterrey, N.L., México
Email: david.gallardofdz@uanl.edu.mx

I. Resumen

Dentro de Schneider-Electric tenemos el área de Schneider Digital y dentro de ese área está el departamento de innovación digital la cual a su vez tiene varios departamentos encargados de mantener, modernizar y soportar diferentes sistemas y aplicaciones, en mi caso soportamos el área de "*Global Supply Chain*" (GSC). Esta área tiene muchas aplicaciones antiguas y modernas por lo que el equipo esta compuesto por personas con muchos años de experiencia en diferentes tecnologías antiguas y modernas que se encargan de mantener el funcionamiento entre estas diferentes aplicaciones a la vez que apoyamos en la modernización de las mismas. El soporte de nivel 2 y 3 es parte de las actividades de este equipo sin embargo, por todo lo mencionado anteriormente, la cantidad de información es mucha debido a que a la cantidad de aplicaciones, servidores, bases de datos, etc. lo cual hace que incluso para una persona con varios años de experiencia la curva de aprendizaje sea muy grande, incluso después de varios años podría no estar familiarizado con todo si a esto le agregamos el tiempo que se tiene que dedicar a dar soporte a cosas que muchas veces tiene que ver con otros equipos como redes, administradores bases de datos, administradores de servidores e infraestructura, etc. hace que la innovación y la modernización se retrase.

Mi propuesta es crear un Chatbot que pueda servir tanto al equipo como a los usuarios. El chatbot podría ayudar a los usuarios a resolver sus problemas y a involucrar a las personas correctas para la resolución de los problemas de soporte. También podría ayudar al equipo a encontrar más rápido la información que necesitan para resolver los problemas ya que podría agilizar la consulta de información. Para entrenar al Chatbot se podría usar la información del sistema de tickets y las notas de los integrantes del equipo. Esto también permitiría reducir la curva de aprendizaje de los nuevos integrantes del equipo para que empiecen a ser productivos más rápidamente.

Para construir este chatbot y solucionar la problemática descrita anteriormente usaré la información del sistema de "*Tickets*" para obtener los datos para el entrenamiento del modelo. También haré el pre-procesamiento y procesamiento de texto con la librería NLTK, creación y entrenamiento de modelos de Machine Learning con Keras y TensorFlow y programación con Python 3.

II. Introducción y antecedentes

Un chatbot es un programa informático que simula y procesa una conversación humana (ya sea escrita o hablada), lo que permite a los humanos interactuar con dispositivos digitales como si se estuvieran comunicando con una persona real. Los chatbots pueden ser tan simples como programas rudimentarios que responden a una consulta simple con una respuesta de una sola línea, o tan sofisticados como asistentes digitales que aprenden y evolucionan para brindar niveles crecientes de personalización a medida que recopilan y procesan información. El código se puede encontrar en Github [2].

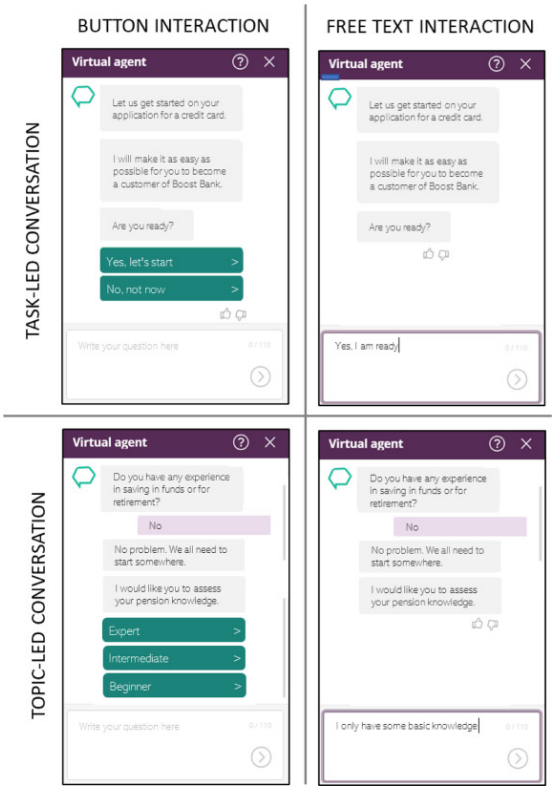

Fig. 1: Tipos de Chatbots. Del lado izquierdo se muestran ejemplos de "*chatbots*" orientados a táreas y del lado derecho los basados en datos y predictivos.

Impulsados por IA, reglas automatizadas, procesamiento de lenguaje natural (NLP) y aprendizaje automático (ML), los chatbots procesan datos para brindar respuestas a solicitudes de todo tipo.

Hay dos tipos principales de chatbots como se muestran en la Fig 1.

Los chatbots orientados a tareas (declarativos) son programas de un solo propósito que se enfocan en realizar una función. Usando reglas, NLP y muy poco ML, generan respuestas automatizadas pero conversacionales a las consultas

de los usuarios. Las interacciones con estos chatbots son altamente específicas y estructuradas y son más aplicables a las funciones de soporte y servicio: piense en preguntas frecuentes sólidas e interactivas. Los chatbots orientados a tareas pueden manejar preguntas comunes, como consultas sobre el horario comercial o transacciones simples que no involucran una variedad de variables. Aunque utilizan NLP para que los usuarios finales puedan experimentarlos de forma conversacional, sus capacidades son bastante básicas. Estos son actualmente los chatbots más utilizados.

Los chatbots basados en datos y predictivos (conversacionales) a menudo se denominan asistentes virtuales o asistentes digitales, y son mucho más sofisticados, interactivos y personalizados que los chatbots orientados a tareas. Estos chatbots son conscientes del contexto y aprovechan la comprensión del lenguaje natural (NLU), NLP y ML para aprender sobre la marcha. Aplican inteligencia predictiva y análisis para permitir la personalización en función de los perfiles de usuario y el comportamiento del usuario en el pasado. Los asistentes digitales pueden conocer las preferencias de un usuario a lo largo del tiempo, brindar recomendaciones e incluso anticiparse a las necesidades. Además de monitorear datos e intenciones, pueden iniciar conversaciones. Siri de Apple y Alexa de Amazon son ejemplos de chatbots predictivos, basados en datos y orientados al consumidor [3].

En la actualidad el uso de chatbots es muy amplio, se usan para seguimiento a solicitudes, captar nuevos clientes, venta de productos y comida y por supuesto soporte técnico [7]. Los chatbots pueden trabajar con sólo texto lo cual implica el poder detectar el sentimiento y el tema del cual se está hablando. Otros chatbots funcionan con opciones o botones para que los usuarios elijan la opción que prefieran, estas opciones pueden ir acompañadas de imágenes u otra información para poder describir mejor la información a los usuarios. Los chatbots también pueden ser mas complejos al grado de poder llamar webservices, crear o modificar registros en bases de datos, crear archivos, ejecutar comandos, etc. lo cual permite que puedan ser muy útiles. Todo esto hace posible que las empresas sean mas productivas, tengan mejores tiempos de respuesta y los empleados pueden usar su tiempo y energías en otras actividades que requieren ser realizadas por personas.

III. Materiales y metodología

Una gráfica de la metodología que se siguió para este proyecto se muestra en la Fig 2 y a continuación se explica con más detalle.

A. Datos

Los datos la parte medular de los proyectos de Machine Learning y en este caso no es la excepción. Para el proyecto se recopiló la información del sistema de "*tickets*" que se utiliza para que los usuarios reciban soporte con los problemas que enfrenten. El sistema de "*tickets*" es en sí una base de conocimientos ya que contiene los problemas y las soluciones que se presentan a lo largo del tiempo. El sistema permite exportar los datos a un archivo CSV. Una vez que se obtiene

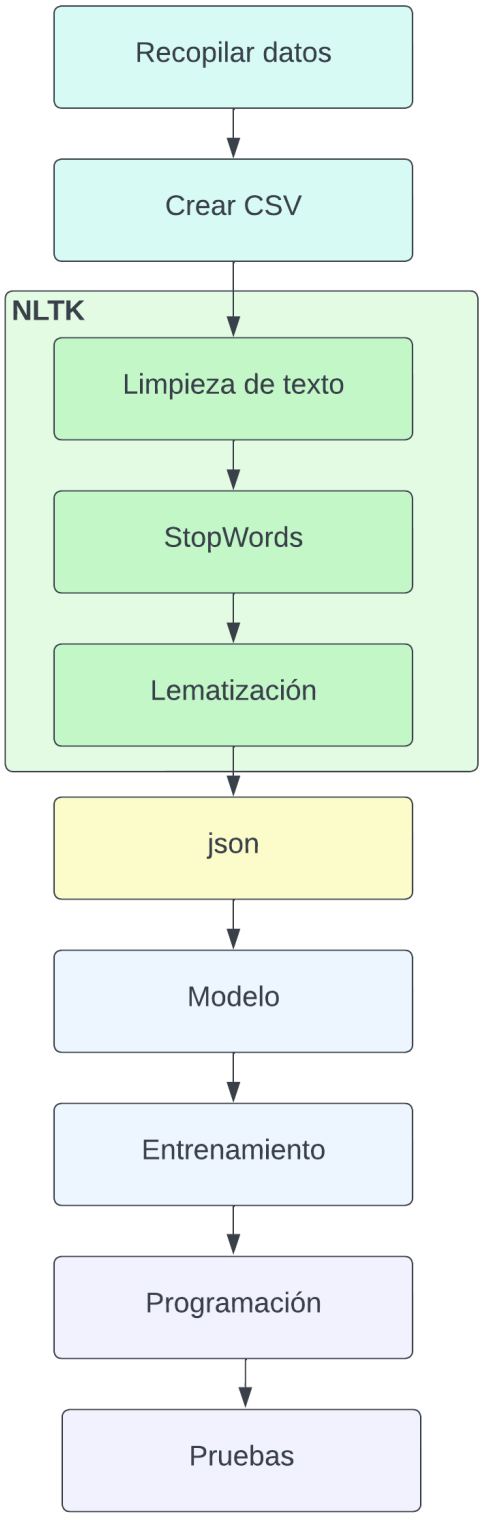

Fig. 2: Metodología.

Resumén del Modelo		
Layer (type)	Output Shape	Param No.
dense (Dense)	(None, 128)	147584
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8256
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 7)	455
Total params: 156,295 Trainable params: 156,295 Non-trainable params: 0		

TABLE I: Resumen del modelo.

este archivo se eliminan las columnas que no son necesarias y se procede al pre-procesamiento del texto.

B. Pre-procesamiento del texto

Antes de poder entrenar el modelo es necesario pre-procesar los datos o el texto contenido en el archivo CSV y para esto se utilizó la librería NLTK. Primero se limpió el texto de símbolos, espacios, saltos de línea, caracteres especiales, etc. y después se removieron las "*StopWords*" para eliminar las palabras que no aportan significado al texto y por ultimo se realizó la Lematización del texto. una vez que se termina el pre-procesamiento del texto tenemos un archivo de texto CSV con los datos limpios y a partir de este se creo un archivo json utilizando una extención de VSCode llamada CSV to JSON [6].

C. Modelo

Se definió un modelo de 3 capas la primera con 128 neuronas y la función de activación "*Relu*", la segunda con 64 neuronas también con la función de activación "*Relu*" y la cantidad de neuronas de la última capa se definió en base a la cantidad de "intenciones" a predecir contenidas en el archivo json en este caso con la función de activación "*Softmax*". Se utilizó el optimizador SGD y ee corrieron 200 "*epochs*". Se obtuvo un accuracy de 0.9894 aunque esto de discutirá más adelante. La tabla I muestra el resumen del modelo que se definió.

D. Programación

El código para el "*Chatbot*" está basado en el código de Jere Xu [1] al cual sólo hice unas pequeñas correcciones para hacerlo funcionar y le agregue código para las métricas y la visualización de las mismas y modifiqué un poco el modelo para poder obtener las métricas y reduje la cantidad de épocas ya que 200 no eran necesarias.

IV. Resultados

En cuanto a los resultados del entrenamiento se muestran muy bien y bastante exactos, prácticamente después de 25 épocas ya no hay muchas mejoras como se muestra en las Figuras 3, 4 y 5 sin embargo ...

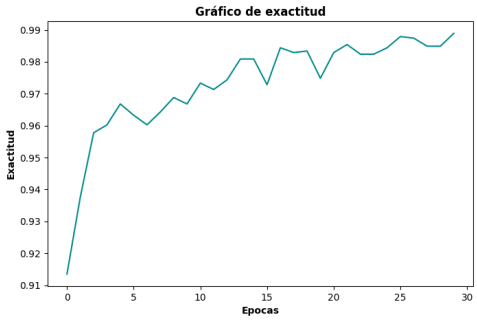

Fig. 3: Gráfico de exactitud.

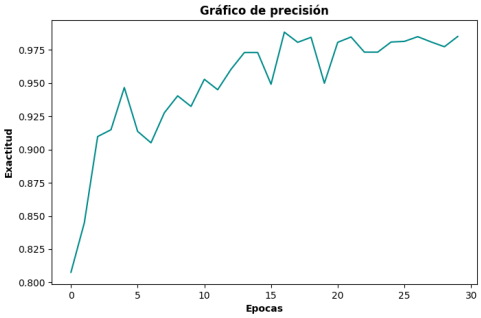

Fig. 4: Gráfico de precisión.

V. Discusión

A la hora de usar el "*chabot*" se puede ver que no es tan efectivo como lo muestran las gráficas y esto es debido a los dos puntos mencionados previamente. En otros casos no puede identificar de que está hablando el usuario cuando usa la palabra "*diskspace*" y regresa un saludo como se puede ver en la Fig. 6 y en esta misma figura se puede ver como si responde bien cuando le damos el nombre de una aplicación como "*Aveva Avantis*" y también lo hace bien cuando usamos la frase "*ship date*" porque sólo tenemos un problema y una solución para esto así que es fácil identificar de que está hablando el usuario.

Me di cuenta de que aunque los temas que se pretenden abarcar con el "*chatbot*" son bien definidos los datos requieren

Fig. 5: Gráfico de pérdida.

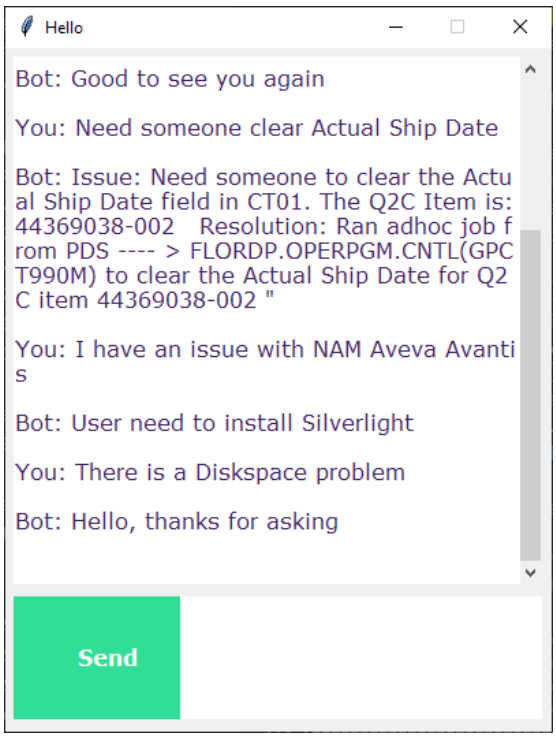

Fig. 6: Chatbot.

un trabajo arduo de pre-procesamiento ya que muchas veces para el mismo problema puede haber diferentes soluciones, por ejemplo, un programa puede fallar debido a que se perdió la conectividad de la red, la base de datos no está respondiendo, hay errores en la información, el servidor tienes problemas, algún servicio no está respondiendo, etc. sin embargo simplemente se va a reportar que el programa X falló, en estos casos es necesario combinar toda la información de este tipo de "*tickets*" en uno solo y que el "*chatbot*" dé como resultado todas las posibles causas de la falla que se han documentado hasta el momento.

Otro caso particular es que como todas las aplicaciones son parte de la división de manufactura es común que se incluyan las mismas palabras en problemas diferentes, en estos casos las "*StopWords*" no ayuda mucho porque solo remueve palabras como artículos, preposiciones, etc. palabras que no tienen un significado objetivo sin embargo en este caso es necesario remover palabras como "*Software, Application, Issue, Application, Manufacturing o Management*" que tienen un significado pero al repetirse tanto dejan de tener valor y sólo agregan ruido a la hora de que el usuario las usa con el "*chabot*".

VI. Conclusiones y trabajo a futuro

Los resultados son bastante prometedores, creo que este proyecto es viable y puede aportar valor al negocio sin embargo el proceso de recopilación y pre-procesamiento de datos necesitar mejorarse para poder dar mejores resultados y para poder lograr esto creo que es necesario invertir mucho tiempo u horas hombre lo cual podría ser un motivo para no seguir adelante con el proyecto.

En caso de seguir adelante con el proyecto sería necesario agregar otras "*StopWords*" que aunque no lo son realmente para este caso es necesario removerlas, también es necesario combinar todos los problemas que tienen diferentes soluciones como se mencionó antes en este documento.

Otra opción sería usar un "*chatbot*" orientado a tareas en donde el usuario vaya seleccionando opciones y respondiendo preguntas para poder ubicar su problema y obtener la solución que necesita ya que los "*chatbots*" basados en datos y predictivos son mas adecuados para casos en los que se desee simular una conversación con alguien.

References

[1] Xu, J. (2022). Simple-Python-Chatbot [Python]. https://github.com/jerrytigerxu/Simple-Python-Chatbot (Original work published 2020)

[2] MCD-PCD/ProyectoFinal at main · degallardo/MCD-PCD. (s/f). GitHub. Recuperado el 15 de julio de 2022, de https://github.com/degallardo/MCD-PCD/tree/main/ProyectoFinal

[3]"What is a Chatbot?" https://www.oracle.com/chatbots/what-is-a-chatbot/ (consultado el 2 de junio de 2022).

[4] Haugeland, I. K. F., Følstad, A., Taylor, C., & Bjørkli, C. A. (2022). Understanding the user experience of customer service chatbots: An experimental study of chatbot interaction design. International Journal of Human-Computer Studies, 161, 102788. https://doi.org/10.1016/j.ijhcs.2022.102788

[5] Rossmann, A., Zimmermann, A., & Hertweck, D. (2020). The Impact of Chatbots on Customer Service Performance (pp. 237–243). https://doi.org/10.1007/978-3-030-51057-2_33

[6] CSV to JSON Converter—Visual Studio Marketplace. (s/f). Recuperado el 22 de julio de 2022, de https://marketplace.visualstudio.com/items?itemName=Chukwuamaka.csvtojson-converter

[7] Rossmann, A., Zimmermann, A., & Hertweck, D. (2020). The Impact of Chatbots on Customer Service Performance (pp. 237–243). https://doi.org/10.1007/978-3-030-51057-2_33