Universitat Oberta de Catalunya

uoc.edu

JacKBot – El chatbot compañero

Nombre Estudiante:

Francisco Domenes Mondragón

Grado de Ingeniería Informática

75.629 - TFG - Inteligencia artificial

Nombre Consultor/a:

Dr. David Isern Alarcón

Nombre Profesor/a responsable de la asignatura:

Dr. Xavier Baró Solé

Fecha Entrega:

26/12/2022

FICHA DEL TRABAJO FINAL

Título del trabajo:	*JacKBot – El chatbot compañero*
Nombre del autor:	*Francisco Domenes Mondragón*
Nombre del consultor/a:	*Dr. David Isern Alarcón*
Nombre del PRA:	*Dr. Xavier Baró Solé*
Fecha de entrega (mm/aaaa):	01/2023
Titulación:	*Grado de Ingeniería Informática*
Área del Trabajo Final:	*75.629 - TFG - Inteligencia artificial*
Idioma del trabajo:	Castellano
Número de créditos:	12
Palabras clave	*Chatbot, NLP, acompañamiento*

Resumen del Trabajo (máximo 250 palabras): *Con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo.*

Según datos del INE en su encuesta única de hogares [1], cerca de cinco millones de personas viven solas en España, ya sea por decisión propia o no, y es un número que va en aumento año tras año y aproximadamente dos millones tienen edades superiores a los 65 años.

Esta soledad que, en situaciones normales, puede ser algo buscado o deseable en algunos casos, para muchas personas, especialmente aquellas en la que la soledad no es una opción elegida, es algo que causa un gran estrés emocional y ante la aparición de problemas graves o situaciones que superen al individuo pueden derivar en problemas como la depresión u otras tipologías mentales.

Mediante técnicas de procesamiento de lenguaje natural (NLP), el lenguaje Python y sus librerías especializadas en Inteligencia Artificial se plantea la creación de un chatbot que permita aliviar al usuario la sensación de soledad, tan presente en nuestra sociedad, y generar una sensación de compañía para hacer frente a esta soledad, ya sea voluntaria o forzosa, para el usuario.

El resultado es un sistema que reconoce las interacciones del usuario y, a través de un archivo de intenciones y una red neuronal, intenta predecir la intención del usuario y proporcionar aleatoriamente una de las respuestas previstas manteniendo una conversación.

No limitar el dominio de trabajo causa que se provoquen algunos fallos en la predicción pero que permiten seguir conversando con el usuario generando la sensación de poder conversar con alguien, que es lo que se buscaba.

Abstract (in English, 250 words or less):

According to the data available on the household survey of the National Statistical Institute [1], near five million people are living alone in Spain, whether if that is their own decision or not. This number is growing every year and approximately two million of them are older than 65 years.

In some situations, this loneliness may be something searched or desired if it is something decided by the subject. But for many other people who have arrived at this situation, the loneliness may be the origin of a lot of emotional stress and, when they are in front of complex issues or situations that overwhelm themselves, these situations could lead to depression or many other mental issues.

Using techniques of Natural Language Processing (NLP), the programming language Python and some of its specialized libraries, our goal is to create a Chatbot that gives the opportunity to the user to ease this feeling of loneliness, so common in the current society, and to bring a feeling of being accompanied to face this situation of solitude.

The result is a system that recognizes the interactions of the user, and through an intents file and a neural network, tries to predict the user intent and provide at random one of the predefined answers maintaining a conversation.

Not limiting the work domain causes some failures in the prediction but it allows to continue conversing with the user, generating the sensation of being able to converse with someone, which is what was sought.

Índice

Contenido

Lista de figuras

Lista de tablas

1 Resumen

Según datos del INE en su encuesta única de hogares [1], cerca de cinco millones de personas viven solas en España, ya sea por decisión propia o no, y es un número que va en aumento año tras año y aproximadamente dos millones tienen edades superiores a los 65 años.

Estos hogares unipersonales se encuentran más desprotegidos frente a la soledad y frente a la capacidad de afrontar situaciones complejas como la reciente pandemia de Covid-19.

Esta soledad que, en situaciones normales, puede ser algo buscado o deseable en algunos casos, para muchas personas, especialmente aquellas en la que la soledad no es una opción elegida, es algo que causa un gran estrés emocional y ante la aparición de problemas graves o situaciones que superen al individuo pueden derivar en problemas como la depresión u otras tipologías mentales.

Mediante técnicas de procesamiento de lenguaje natural (NLP) y el lenguaje Python y sus librerías especializadas en Inteligencia Artificial se plantea la creación de un *chatbot* que permita aliviar al usuario la sensación de soledad, tan presente en nuestra sociedad y generar una sensación de compañía para hacer frente a esta soledad ya sea voluntaria o forzosa para el usuario.

El resultado es un sistema que reconoce las interacciones del usuario y a través de un archivo de intenciones y una red neuronal intenta predecir la intención del usuario y proporcionar aleatoriamente una de las respuestas previstas manteniendo una conversación.

Tras las pruebas realizadas, se comprueba que a causa de no limitar el dominio de trabajo en un área concreta y dejar el abanico de posibilidades de conversación abierto, causa que no se alcancen niveles de acierto de predicción excesivamente elevados, causados también por un volumen no demasiado extenso de datos para el aprendizaje

de la red neuronal, pero que se ha decidido así para tener un mayor control sobre la información que se le facilita al sistema. A pesar de ello, el sistema permite seguir conversando y poder aportar esa sensación de estar conversando con alguien, que es lo que se buscaba como objetivo inicial.

La aportación que se espera conseguir con este modelo elaborado es la creación de un sistema que permita a un usuario mantener una conversación más o menos fluida sin limitarlo a un entorno concreto y basado en una serie de estados concretos y una serie de respuestas predefinidas. Estas respuestas predefinidas nos permiten tener un mayor control sobre lo que va a percibir el usuario y que, aunque a priori pueda ser una limitación, se trata de una ventaja ya que, sin variar el resto de componentes, solamente el cambio de las respuestas predefinidas permite un grado de personalización respecto al usuario que lo va a utilizar y permite un alto grado de adaptación a posibles perfiles de usuarios, como una adaptación a lenguaje para público más juvenil o de edad avanzada, o de un argot específico, sin tener que reentrenar el modelo cada vez. También en estas respuestas predefinidas se realizan preguntas para hacer avanzar el flujo de la conversación, por lo que solamente modificando estas preguntas se puede reorientar el flujo establecido visitando los estados creados en órdenes más específicos o adecuados a situaciones concretas o usuarios específicos.

2 Introducción

2.1 Contexto y justificación del Trabajo

Con este escenario tan aciago para un alto porcentaje de la población se presenta la necesidad de crear un sistema que pueda paliar, en la medida de lo posible, esta sensación de soledad a la que se enfrentan estas personas. Este sistema no sería un servicio de ayuda o información, de los que ya existen algunos en el mercado, como el servicio “Te acompaña” de Cruz Roja [2] u otros sistemas que proporcionan servicios de seguros médicos o compañías de alarmas o reservas de billetes y que cada vez es más común encontrar en los servicios web de muchas compañías que ofrecen algún producto al usuario.

Lo que se pretende en este trabajo es que estas personas puedan tener simplemente alguien con quien hablar y a la que explicarle sus cosas o que simplemente las escuche. Con esta filosofía se propone la creación de JacKBot, un acompañante virtual que pueda proporcionar a todo aquel que se siente solo en algún momento de su vida, que pueda tener alguien con quien mantener una conversación y, de ese modo, aliviar un poco esa sensación de soledad que en ocasiones puede ser insoportable para quien la sufre.

Gracias a la inteligencia artificial, se propone generar un sistema que consista en un *Chatbot* que pueda mantener una conversación genérica con el usuario sin más pretensiones que la de acompañarle frente a la soledad y que le proporcione una sensación de compañía que le permite sobreponerse a esa sensación de estar solo y no tener alguien en quien apoyarse.

2.2 Objetivos del Trabajo

Como ya se ha indicado, el propósito de este sistema no es el de un asistente virtual para conectar al usuario con el resto del mundo, como, por ejemplo, Alexa de Amazon [3], ni la de información o ayuda como la citada de la Cruz Roja [2].

El propósito que aquí se busca es mucho más modesto y solamente pretende generar una sensación de acompañamiento en el usuario y que sienta que alguien le escucha y que dispone de un compañero virtual con el que hablar, o chatear en este caso.

Algunos de los objetivos que nos marcamos se detallan a continuación.

Objetivos principales:

- Aplicar los conocimientos adquiridos durante el grado y, mediante las herramientas de inteligencia artificial y el lenguaje Python, llegar a generar un *Chatbot* operativo.

- Construir un flujo de estados y de intenciones que permitan al asistente mantener una conversación coherente con las necesidades del usuario.

- Optar por una estructura de creación del *Chatbot* que permita un diseño sencillo pero eficiente que pueda ser ágil y rápida.

Objetivos secundarios:

- Generar un *Chatbot* que mantenga una conversación genérica con el usuario que mediante preguntas y respuestas facilite una sensación de acompañamiento en la persona.

- Ayudar a minimizar la sensación de soledad en las personas que viven solas, o aquellas que aun viviendo con más gente puedan tener la sensación de sentirse solas.

- Generar un sistema que permita, en un futuro, ampliar sus capacidades con modelos generativos (como *Bloom* de *BigScience* de *HuggingFace* [4]), o incluso, reconocimiento de voz.

2.3 Enfoque y método seguido

La gran expansión del lenguaje de programación Python y su uso generalizado en los proyectos de Inteligencia Artificial han conseguido que haya disponibles para el usuario final multitud de proyectos en código abierto e incluso modelos funcionales para la creación por cualquier persona de un *chatbot* en pocos pasos y, en algunos casos, incluso con pocos conocimientos de programación como Rasa [5]. Además, también se han generado multitud de librerías específicas para este lenguaje de programación y fuertemente orientadas hacia las posibles aplicaciones de la Inteligencia Artificial como las orientadas al aprendizaje automático u otras tareas o incluso para la generación automática de *chatbots* como *Chatterbot* [6].

A pesar de que sería posible coger alguno de los proyectos ya operativos que existen y adaptarlo a las necesidades específicas de este trabajo, se ha optado por seguir una estrategia basada en la creación de un sistema desde cero. Este enfoque, además de permitir una personalización más específica para la tarea escogida, implica un mayor control del proceso completo de creación y ayuda al estudiante a tener una visión más clara de las herramientas que participan en toda la creación y ayuda al desarrollo personal y a mejorar las capacidades que son necesarias para su creación.

Básicamente, se seguirá una metodología en cascada, como se verá en el siguiente apartado, realizando las tareas una tras otra, aunque, especialmente en las tareas de desarrollo, que se mezclará con una metodología iterativa y retroalimentando ambas fases, ya que puede suceder que cuando apliquemos el modelo a las especificaciones creadas, no se adapten al modelo final y se tengan que redefinir dichas especificaciones o se hayan de modificar y volverlas a aplicar al modelo generado. En el caso de las subtareas de cada fase, se podrá producir solapamiento en algunas de las tareas, sobre todo en aquellas que comportan la elaboración y entrega de las diferentes PEC ya que se realizarán simultáneamente con otras subtareas, ya que muchas veces están relacionadas.

2.4 Planificación del Trabajo

Siguiendo el enfoque que se ha comentado en el punto anterior, se basará el desarrollo del producto que se va a trabajar en la utilización del lenguaje de programación Python y de algunas de entre la multitud de librerías especializadas que nos ofrece como pueden ser *Sklearn* [6] o *Spacy* [7].

Gran parte del resultado final del *chatbot* se basará en una definición correcta de los estados y las respuestas que son necesarias para el correcto funcionamiento de la herramienta, esta tarea será en la que más tiempo se empleará, dado que, en gran medida, de su definición dependerá el posterior rendimiento del sistema, y es muy probable que una vez implementado en el sistema se tengan que hacer posteriormente modificaciones y reajustes que impliquen una mayor dedicación.

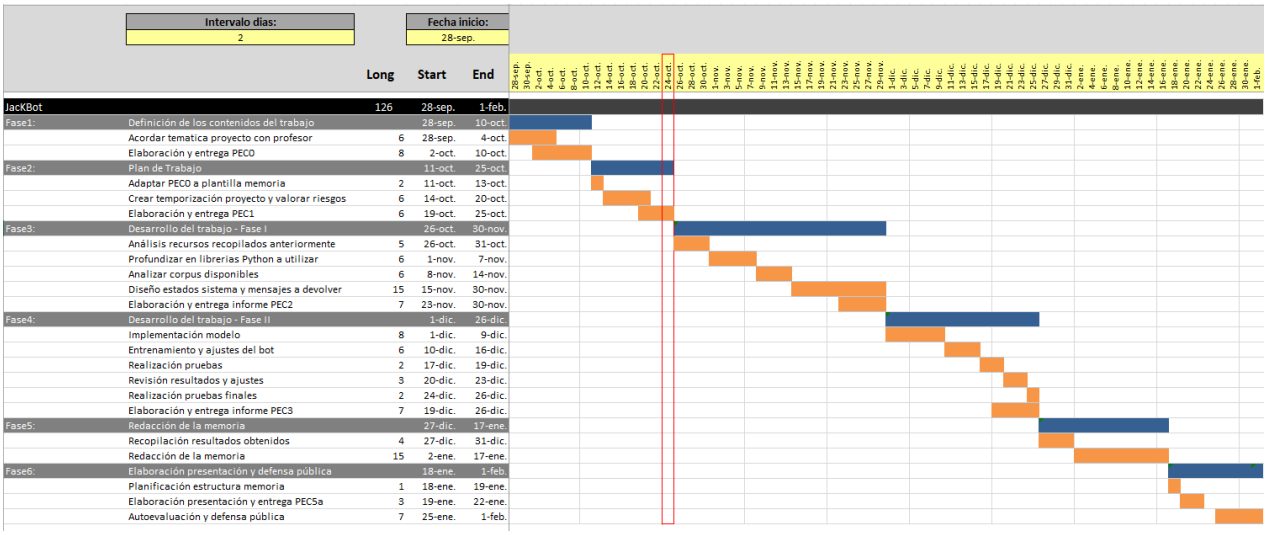
A continuación, se mostrará una planificación inicial prevista con los tiempos a emplear en las diferentes tareas que vamos a definir. Se ha de indicar que la planificación indicada se reparte a través de periodos de semanas y días, aunque la dedicación en las fechas indicadas se realizará de un modo irregular debido a la compaginación de éstas con la jornada laboral y otras ocupaciones, por lo que los momentos de mayor dedicación dentro de los presentados serán especialmente en tardes y fines de semana. En próximas entregas durante el desarrollo del proyecto, se realizará un análisis del seguimiento de dicha planificación para revisar el cumplimiento o no del mismo y las posibles desviaciones.

Las principales fases que formarán parte del proceso de desarrollo de la herramienta, y que se distribuirán para que coincidan con las fechas establecidas para las diferentes entregas parciales de la asignatura a la que se adscribe, son las siguientes:

Fases definidas	Fecha Inicio	Fecha Fin
Definición de los contenidos del trabajo	28/09/2022	10/10/2022
Plan de Trabajo	11/10/2022	25/10/2022
Desarrollo del trabajo - Fase I	26/10/2022	30/11/2022
Desarrollo del trabajo - Fase II	01/12/2022	26/12/2022
Redacción de la memoria	27/12/2022	17/01/2023
Elaboración presentación y defensa pública	18/01/2023	01/02/2023

Tabla 1: Listado de tareas a realizar

A continuación, se presentará un diagrama de *Gantt* con el desglose de estas tareas iniciales en las subtareas que la formarán. Posteriormente se definirán con mayor detalle dichas tareas y subtareas y se definirán los posibles riesgos que es posible que aparezcan durante el proceso.



		Long	Start	End
JacKBot		126	28-sep.	1-feb.
Fase1:	Definición de los contenidos del trabajo		28-sep.	10-oct.
	Acordar tematica proyecto con profesor	6	28-sep.	4-oct.
	Elaboración y entrega PEC0	8	2-oct.	10-oct.
Fase2:	Plan de Trabajo		11-oct.	25-oct.
	Adaptar PEC0 a plantilla memoria	2	11-oct.	13-oct.
	Crear temporización proyecto y valorar riesgos	6	14-oct.	20-oct.
	Elaboración y entrega PEC1	6	19-oct.	25-oct.
Fase3:	Desarrollo del trabajo - Fase I		26-oct.	30-nov.
	Análisis recursos recopilados anteriormente	5	26-oct.	31-oct.
	Profundizar en librerias Python a utilizar	6	1-nov.	7-nov.
	Analizar corpus disponibles	6	8-nov.	14-nov.
	Diseño estados sistema y mensajes a devolver	15	15-nov.	30-nov.
	Elaboración y entrega informe PEC2	7	23-nov.	30-nov.
Fase4:	Desarrollo del trabajo - Fase II		1-dic.	26-dic.
	Implementación modelo	8	1-dic.	9-dic.
	Entrenamiento y ajustes del bot	6	10-dic.	16-dic.
	Realización pruebas	2	17-dic.	19-dic.
	Revisión resultados y ajustes	3	20-dic.	23-dic.
	Realización pruebas finales	2	24-dic.	26-dic.
	Elaboración y entrega informe PEC3	7	19-dic.	26-dic.
Fase5:	Redacción de la memoria		27-dic.	17-ene.
	Recopilación resultados obtenidos	4	27-dic.	31-dic.
	Redacción de la memoria	15	2-ene.	17-ene.
Fase6:	Elaboración presentación y defensa pública		18-ene.	1-feb.
	Planificación estructura memoria	1	18-ene.	19-ene.
	Elaboración presentación y entrega PEC5a	3	19-ene.	22-ene.
	Autoevaluación y defensa pública	7	25-ene.	1-feb.

Ilustración 1: Diagrama de Gantt previsto

Se describen a continuación el contenido de cada una de las tareas y subtareas que hemos visto en el diagrama realizado:

- Fase1: Definición de los contenidos del trabajo. En esta primera fase, se definen las características y razones que han llevado a la decisión de la elaboración del presente proyecto, así como los objetivos primarios y secundarios que se pretender alcanzar. Las subtareas que se plantean son:

- Acordar temática proyecto con profesor. Se revisan los recursos que se han ido obteniendo previamente y que han ayudado a tomar la decisión de optar por la presente opción y se presentan al responsable para que confirme la idoneidad de éste o no.

- Elaboración y entrega PEC0. Se inicia la redacción de la información que se incluirá en esta entrega. Una vez recibida esta información, se establecen los objetivos, la razón y alcance que se pretende obtener, se documenta y se realiza la entrega correspondiente. Esta subtarea se compagina con la anterior parcialmente.

- Fase2: Plan de trabajo. En esta fase se pretende definir el calendario de trabajo que se seguirá para tratar de lograr los objetivos marcados previamente y poder obtener un resultado satisfactorio. Las subtareas que se plantean son:

  - Adaptar PEC0 a plantilla memoria. Se integra el documento realizado en la entrega anterior a la estructura definitiva de acuerdo con las indicaciones recibidas. Realmente, esta tarea se realiza compaginada con la siguiente, una vez se reciben las valoraciones de la entrega anterior, pero se decide mostrarla como tarea previa de cara a una mayor claridad en el documento y que sería una tarea que haber realizado con anterioridad a la siguiente.

  - Crear temporización proyecto y valorar riesgos. Aquí se plantea la forma en que se va a estructurar temporalmente el proyecto, definiendo las diferentes tareas a realizar y su posición en el diagrama de tiempo. Una vez realizada la distribución, se detectan los posibles riegos que podrían afectar al cumplimiento del calendario propuesto para incluirlos en el documento final.

  - Elaboración y entrega PEC1. Una vez establecidos los rangos temporales necesarios, se inicia la redacción del documento a entregar con el correspondiente diagrama de *Gantt* y las valoraciones de riesgos encontrados.

- Desarrollo del trabajo – Fase I. En esta fase, se inicia propiamente el desarrollo del sistema que se va a implementar, analizando las herramientas disponibles y aquellas que se van a utilizar en el proyecto final. Esta es la fase más importante del proyecto, ya que el correcto funcionamiento del resultado final dependerá de establecer una correcta distribución de los estados a través de los pasará el sistema, de las respuestas que ha de ofrecer el producto y de la correcta capacidad que tenga para entender lo que el usuario le indique. Las subtareas previstas para esta fase del trabajo son:

    - Análisis recursos recopilados anteriormente. Se ha de hacer una recopilación de información que se ha obtenido anteriormente para detectar de entre todas las posibilidades las más adecuadas para el presente proyecto.

    - Profundizar en librerías Python a utilizar. Se analizarán en más detalle, de todas las posibilidades que se han detectado, aquellas que se han optado por utilizar para detectar los requerimientos necesarios y su utilización correcta. Esta previa profundización nos permitirá tener un mayor conocimiento de estas y ayudará a agilizar los pasos posteriores que requieren un esfuerzo de programación mayor.

    - Analizar corpus disponibles. Se analizarán los diferentes sets de datos captados del leguaje natural que se encuentran disponibles en la red para ver cual se adapta mejor al producto final. Finalmente se decidirá por la creación de un set de datos a propósito para el sistema a crear.

    - Diseño estados sistema y mensajes a devolver. Esta es la parte más importante del proyecto, ya que es la que marcará las respuestas que otorgará el sistema y el que puede determinar un correcto rendimiento de este o no, y por ello, es a la que se ha asignado una cantidad mayor de tiempo.

    - Elaboración y entrega informe PEC2. Se compaginará la elaboración de esta entrega con las subtareas que se van realizando, en ella, se analizará el cumplimiento de la planificación prevista con la real y se detectarán las posibles desviaciones que se puedan producir.

- Desarrollo del trabajo – Fase II. Una vez se tienen los elementos principales, se procederá a la implantación y pruebas del modelo, realizando los ajustes necesarios para tratar de lograr el mejor rendimiento posible. Las subtareas que se incluyen en esta fase son:

    - Implementación del modelo. Se realizará una primera implementación del modelo para poder empezar a realizar las pruebas necesarias y detectar posibles mejoras o fallos que se puedan producir para solucionarlas antes de la entrega final.

    - Entrenamiento y ajustes del bot. Tras la implementación, se realizará el entrenamiento del sistema y se empezarán a realizar ajustes sobre el mismo según los resultados obtenidos.

    - Realización de pruebas. Se realizarán pruebas sobre el sistema para comprobar el rendimiento. Al tratar de obtener un *chatbot* que acompañe al usuario la valoración de los resultados será subjetiva y se tratará de realizar sobre pruebas con otros usuarios u otras pruebas, en medida de las posibilidades temporales que se tengan.

    - Revisión resultados y ajustes. Se revisarán los resultados de las pruebas realizadas que se hayan podido ejecutar o las internas que se realicen y se harán los ajustes necesarios que haga falta.

    - Realización pruebas finales. Tras los ajustes realizados se realizará alguna prueba para comprobar si el resultado ha sido satisfactorio. Se ha de tener en cuenta que en la planificación este periodo coincide con los días del periodo navideño por lo que se prevé que el tiempo disponible sea menor.

    - Elaboración y entrega informe PEC3. Durante el desarrollo de las pruebas y los ajustes se procederá al redactado del informe para comprobar la posible desviación del mismo respecto a la planificación realizada y el resultado obtenido. Como en el punto anterior, en este periodo coincide parte del periodo navideño por lo que, aunque se han planificado varios días, los días reales de implicación serán menores por lo que se reajustarán las horas

dedicadas en el resto de los días planificados para minimizar la posible desviación.

- Redacción de la memoria. Durante esta fase se realizará el redactado final de la memoria que resumirá todo el trabajo realizado en este periodo para incluir los resultados y conclusiones a los que se hayan llegado. Las subtareas incluidas son:

    - Recopilación resultados obtenidos. Se analiza toda la información recogida a lo largo del proyecto para detectar la información importante que contendrá la memoria final.

    - Redacción de la memoria. Se realiza el documento final que comportará el resumen de todo lo elaborado durante el proyecto para su presentación final en la entrega correspondiente a este periodo.

- Elaboración presentación y defensa pública. Una vez finalizado el proyecto y entregada la presente memoria, se preparará la presentación que se realizará para la defensa pública del proyecto delante del jurado asignado. Las subtareas que se realizarán serán:

    - Planificación estructura memoria. Se analizará la memoria realizada para detectar los puntos importantes que tendrán que aparecen en la presentación final que se realice.

    - Elaboración presentación y entrega PEC5a. Con la información recopilada se preparará la presentación final que servirá de guía para la defensa pública que se ha de realizar.

    - Como última tarea, se ejecutará la presentación de la defensa pública y se realizará la autoevaluación correspondiente al cierre del proyecto.

De entre todas las tareas a realizar, se consideran como más prioritarias las del diseño de estados del sistema y mensajes a devolver y la de implantación y entrenamiento del sistema, ya que son las necesarias para la obtención del producto final, en caso de que

fuera necesario un ajuste de tareas se podrían limitar otras para asegurar el cumplimiento de estas.

Una vez finalizada la planificación, se han detectado los posibles riesgos que se pueden producir a lo largo del desarrollo y que pueden implicar el no cumplimiento de esta. Entre los principales riesgos detectados podemos encontrar:

- Comportamiento erróneo del sistema obtenido. Se puede dar la situación de que el desarrollo que se realice de situaciones y respuestas no de un buen resultado final y que en posteriores ajustes no se consiga ajustar para que se comporte como se espera. Una posible mitigación a este riesgo sería minimizar los estados posibles y reducir las expectativas iniciales a un modelo más limitado y centrado en pocos estados y unas respuestas más limitadas.

- Set de datos de entrada limitado o entrenamiento insuficiente. Si no se logra un volumen de datos correcto o el entrenamiento no es suficiente puede dar lugar a que el sistema se vea incapacitado a obtener los resultados esperados. Una posible mitigación sería la de la utilización de un *chatbot* ya preconstruido y ajustarlo a las necesidades requeridas.

2.5 Breve sumario de contribuciones y productos obtenidos

Los principales productos utilizados y que obtenemos al final del desarrollo son los siguientes:

- intents.json: Este es el archivo que se genera al inicio del desarrollo y que consta de tres partes: los patrones, que son las sentencias que usaremos como parte del entrenamiento del modelo y que formarán la base a partir de la cual se realizarán las predicciones sobre las intenciones de las interacciones que realice el usuario; las etiquetas, que corresponden a la identificación de los estados en los que se puede hallar el sistema y que es aquello que pretenderá predecir el sistema según lo que introduzca el usuario; y las respuestas, que será el

conjunto de posibles respuestas a cada estado y que serán seleccionadas aleatoriamente según el estado detectado y que incluyen, en algunos casos, preguntas que orientarán la siguiente interacción del usuario hacia el estado que se haya previsto en el flujograma establecido.

- lemmas.pkl: Este archivo se genera con el diseño del modelo y contendrá el listado de palabras referenciales ordenadas y sin duplicados que puede detectar el sistema, según lo que se le ha introducido, para su uso en el interfaz.
- Etiquetas.pkl: También generado al mismo tiempo que el anterior, consta del listado de etiquetas que se encontraban en el archivo de intenciones ordenadas y sin duplicados, para su uso en el interfaz.
- jackBot_model.ipynb: Contiene la parte del diseño en Python realizado por medio de la plataforma de programación *Jupiter Notebook* [8], para el tratamiento del conjunto de datos inicial y todo el diseño del modelo de la red neuronal que utilizaremos para las predicciones.
- JackBot_model.h5: Contiene el modelo que se ha creado en el archivo anterior ya entrenado y ajustado, listo para importar y ser utilizado en el archivo siguiente que será el que generará el interfaz y definirá las interacciones.
- jackBot_Chatbot.ipynb: Contiene la parte del diseño en Python que importa el modelo que acabamos de crear y los archivos de lemas y etiquetas para poder tratar las interacciones del usuario y poder entregarlas al modelo que ya tenemos entrenado para realizar las predicciones y dar respuesta. También incluye la interfaz gráfica que será sobre la que interactuará el usuario.

2.6 Breve descripción de los otros capítulos de la memoria

A continuación, mostramos un breve resumen del contenido de cada uno de los capítulos que podemos encontrar en la memoria:

- Capítulo 1. Resumen. Muestra una breve introducción que presenta las razones que han llevado a la elaboración de este trabajo.

- Capítulo 2. Introducción. En este capítulo y a través de sus diferentes subapartados establecemos los fundamentos del trabajo, su justificación, el enfoque a utilizar, los objetivos que se pretende cumplir y la planificación que se prevé realizar para poder llevar a cabo las tareas previstas.

- Capítulo 3. Estado del arte. Breve resumen de la situación actual de la problemática que se intenta resolver y la relación del proyecto a realizar con esta problemática.

- Capítulo 4. Metodología. En este capítulo se hará una descripción de las metodologías que se hayan utilizado durante el desarrollo del proyecto.

- Capítulo 5. Resultados. Aquí se presentarán los resultados obtenidos a lo largo del proyecto en correspondencia con los métodos utilizados y las experiencias adquiridas en su utilización.

- Capítulo 6. Discusión. Aquí se ponen en contraposición los resultados obtenidos con los que se pretendían obtener dentro del contexto del presente proyecto.

- Capítulo 7. Conclusiones. Se presentarán las conclusiones obtenidas a lo largo del trabajo, así como una valoración entre lo que se ha obtenido y lo que se pretendía y las razones que esta valoración obtenga. También se valorará el seguimiento de la planificación realizada y la metodología seguida para analizar si ha sido una elección acertada o se tendría que haber seguido otra línea de desarrollo. Además, se introducirán unas líneas de futuro sobre posibles evoluciones que pueda tener el modelo para poder desarrollar todas sus posibilidades.

- Capítulos 8 y 9. Glosario y Bibliografía. Incluirán la lista de términos utilizados en el primer capítulo y un listado de las referencias utilizadas, ya sea a nivel de consulta o por referencias explícitas en el documento.

- Capítulo 10. Anexos. En caso de que durante el desarrollo del proyecto se considere necesario la inclusión de otra información más extensa, pero de necesario conocimiento para entender el proyecto, pero que no formen parte intrínseca de la memoria se incluirán en este apartado.

3 Estado del arte

El mundo de la Inteligencia artificial se halla cada vez más presente en nuestras vidas, a través de sistemas que nos hacen recomendaciones sobre prácticamente todo, sistemas de detección de objetos, incluso sistemas que crean imágenes, escriben o programan solos, como las últimas aplicaciones en código abierto de la compañía *OpenAI* [9] basadas en el modelo GPT-3 de su creación que, en los últimos tiempos han tenido mucha repercusión como DALL-E o ChatGPT.

Una de las aplicaciones que también están muy extendidas son los *chatbots* que son sistemas que interactúan con el usuario como si este se encontrara hablando con una persona en lugar de con una máquina. Estos sistemas se basan en modelos de procesado del lenguaje natural o NLP que entienden lo que el usuario quiere decir y le proporcionan una respuesta acorde a lo que se le ha solicitado, ya sea por medio de reglas, respuestas predeterminadas o incluso respuestas autogeneradas.

La dificultad en estos sistemas es la ilimitada posibilidad de temas de los que le puede consultar un humano y que haría completamente imposible que pudiera abarcarlos todos. Por ello, los sistemas actuales, se basan en su mayoría en unos modelos muy enfocados a tareas concretas, como las reservas de hoteles, asistencia técnica, información de productos u otras posibilidades. Ejemplos de este tipo de *chatbots* enfocados puede ser el Trabajo de fin de grado de Anesti Álvarez Ruiz sobre consultas en el proceso de preinscripción escolar en el área de Barcelona [10] o trabajos como el de Noelia Martinez Diaz sobre un *chatbot* y una web para identificar sonidos del cielo [11], este último se enfoca en la utilización de una plataforma específica para creación de *chatbots* como es Rasa [5].

Dentro de estos modelos, existen algunos más genéricos y que se basan en la interconectividad con el mundo y que permiten realizar muchas funciones directamente con el asistente, como podría ser Alexa de Amazon [3] u otros sistemas similares.

La gran expansión del lenguaje de programación Python, el desarrollo de multitud de librerías de este lenguaje especializadas en la Inteligencia Artificial y la presencia en la

red de cada vez más modelos en código abierto disponibles para todo el mundo han ayudado al crecimiento de estos modelos y generalización de uso en multitud de aplicaciones. Además, han proliferado para este lenguaje de programación multitud de librerías enfocadas en esta área e incluso específicas para la creación de *chatbots* como la ya mencionada *Chatterbot* [6].

Pero como ya hemos indicado, la mayoría de estos modelos se encuentran muy enfocados a tareas concretas y otros solamente se encuentran en inglés, ya que es el más extendido y con mayor número de desarrolladores y aunque cada vez más están apareciendo más recursos en español y están evolucionando como por ejemplo la librería *Spacy* [7] que dispone de modelos específicos en este idioma.

Con este trasfondo, es donde se ha detectado que, como se indicaba al inicio de la presente memoria, uno de los problemas de nuestra sociedad actual es la soledad, y en muchas ocasiones, lo que menos se necesita es una interconexión con toda la red o el acceso a multitud de servicios, sino que lo que se busca es simplemente tener una entidad con la que poder conversar un rato y que ayude a disminuir esta sensación puntual de soledad que cualquiera puede sentir en algún momento.

Por ello, este sistema solamente pretende ser una entidad que en estas ocasiones pueda ofrecer alguien con quien conversar un momento, sin necesidad de interconexión con el exterior no que te ofrezca servicios ni otras cosas, solamente con la pretensión de ser una entidad que pueda ofrecer un poco de compañía y alguien que te escuche. Una compañía que ha de ser entendida como el hecho de tener alguien con quien conversar o a quien explicarle las cosas cuando se llega a casa, más concretamente, se centra en lograr la sensación de tener a alguien que te escucha, algo tan simple y complejo a la vez.

4 Metodología

El desarrollo se realiza en dos grandes bloques: en el primero se diseña el conjunto de estados y las respuestas que ofrecerá el sistema, además del conjunto de patrones que formarán parte del conjunto de datos que se utilizarán como base de aprendizaje. En el segundo bloque realizaremos propiamente el diseño del modelo que se dividirá en una parte que gestionará los datos, generará el modelo, lo entrenará y producirá una exportación de un modelo entrenado que en la segunda parte se importará en otro módulo desde el que se realizarán las predicciones y se definirá la interfaz gráfica donde estas se llevarán a cabo.

Como se podrá ver a continuación, la base de todo el modelo es el archivo de intenciones. El mismo modelo, con variaciones en este archivo puede ser utilizado de diferentes modos. Modificando las etiquetas y las preguntas que se realizan dentro de las respuestas, permitirá modificar y reorganizar el flujograma establecido. Modificando el contenido de etiquetas, patrones y respuestas, se pueden adaptar a otros dominios de trabajo definiendo un nuevo conjunto de patrones, unas nuevas etiquetas, y unas respuestas que incluyan preguntas que dirijan la conversación y establezcan un nuevo flujograma, que si se aplica a dominios cerrados o más específicos puede tratarse de un flujograma mucho más estricto que el establecido para el diseño actual.

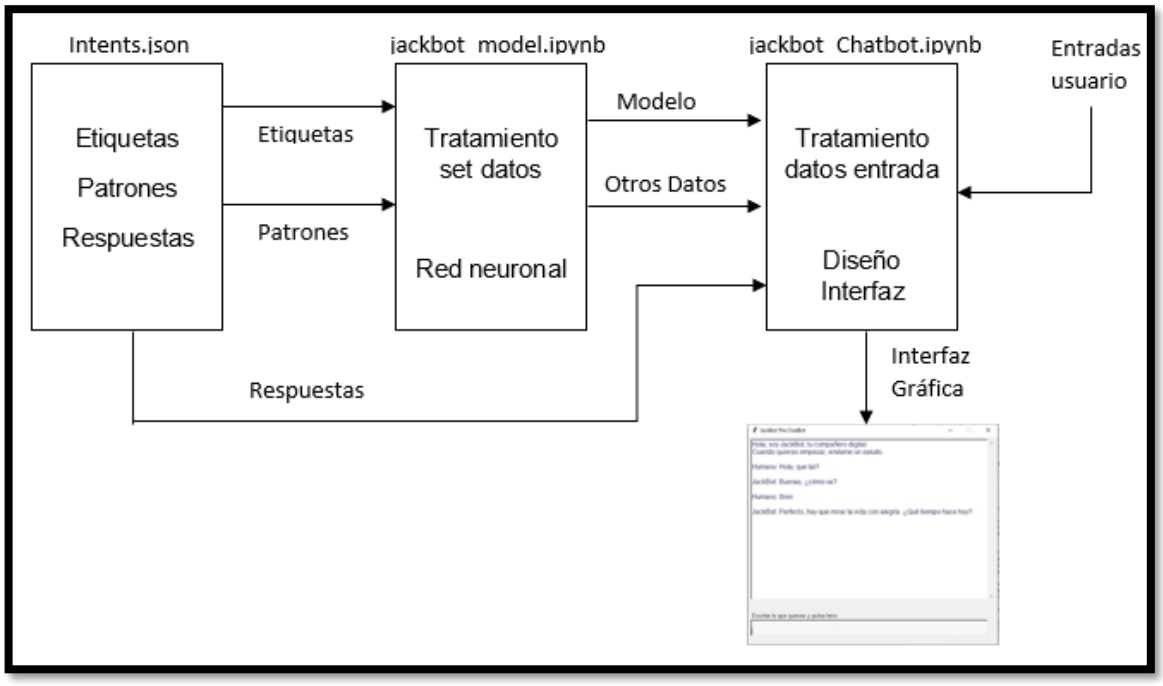


Ilustración 2.Estructura del modelo

4.1 Definición de estados, patrones y respuestas (archivo intents.json)

Primero se verá la estructura del archivo de mensajes o intenciones que es la parte más importante del diseño, ya que, cuanto mayor sea el conjunto de datos, mejor se podrán realizar las predicciones posteriores. A pesar de que existen otros conjuntos de datos preconstruidos, su elección implicaría una revisión y etiquetado, aunque aportaría un conjunto de datos mayor. Por lo que se decide la creación completa del conjunto de datos, para poseer un mayor control de la información que se le da al sistema y la que devolverá, y una mayor implicación del desarrollador en el modelo, a pesar de que ello implique disponer de un conjunto menor de datos y un rendimiento menor del sistema.

Este conjunto de datos inicial, en concreto, las agrupaciones de patrones y etiquetas formarán el corpus o conjunto de datos tomados del lenguaje natural y etiquetados que se utilizarán como datos de entrada y entrenamiento para la red neuronal, donde los patrones serán el conjunto de datos que se quiere que el sistema aprenda a reconocer, y las etiquetas que representarán el estado o, como indica su nombre, la etiqueta que resume lo que se quiere detectar y predecir. Finalmente, el conjunto se compone de un total de 450 frases agrupadas en 34 etiquetas o estados y que se componen de 560 lemas o palabras en forma canónica diferentes.

Si bien este conjunto de datos presenta una limitación en cuanto a los datos de entrada, también lo dota de un alto grado de control ya que permite tener limitadas las posibles respuestas y evita caer en comportamientos inadecuados. Además, también proporciona un alto grado de flexibilidad, ya que una modificación en el archivo, como el aporte de más volumen de datos puede aumentar su fiabilidad y una modificación en el conjunto de datos, como se ha comentado, permite variar su dominio de acción o su adaptación a otras necesidades.

Tanto los estados del sistema como los mensajes a devolver se incluyen dentro de un archivo denominado “intents.json” escrito en formato *JSON* [12] y del que se incluirá un ejemplo de su estructura en el apartado de anexos de la presente memoria.

En este archivo, se define la información en tres grandes bloques:

- Etiquetas: Definen las intenciones a detectar y serán las que marquen los diferentes estados a los que ha de hacer frente el sistema.
- Patrones: Son la base que enseñará a nuestro modelo a detectar e interpretar las posibles interacciones que puede presentar el usuario con el sistema y que constituyen además la base del corpus o conjunto de datos utilizado para el entrenamiento de éste.

- Respuestas: Son los mensajes que se han definido como posibles mensajes que devolverá el sistema, una vez detectada la intención de la interacción del usuario, además, en muchos casos, incluyen preguntas que ayudarán a orientar la conversación para poder seguir el flujograma establecido.

Los posibles estados en los que se puede encontrar el sistema se dividirán en dos grandes grupos. Los estados jerarquizados, en los que se establecerá una estructura clara de los diferentes estados y que, mediante las preguntas que se incluirán en las respuestas, permitirán al sistema establecer un guion básico de desarrollo de la conversación por el que guiar al usuario. Los estados libres, serán una serie de estados que no se encontrarán definidos en la jerarquía establecida y que no tienen relación con el resto, pero que permitirán ampliar el rango de respuestas del sistema frente a interacciones del usuario que se salgan de la previsión realizada.

Con esta separación de estados se pretende lograr un progreso de la conversación coherente, pero que esté preparado ante eventualidades no previstas pero que puedan ser gestionadas correctamente. También pretenden conseguir que el sistema no sea tan rígido como otros sistemas más enfocados a tareas específicas como los de atención al cliente de un producto específico y que permitan enfrentarse a conversaciones más casuales y con un progreso más imprevisible. En los estados jerárquicos se pretende dar un poco de aleatoriedad al sistema introduciendo en cada estado, de las respuestas aleatorias posibles varios estados destino para de este modo alternar los posibles recorridos del diálogo final y hacerlo un poco más aleatorio. Esto se verá en el diagrama de estados que se incluirá más abajo tras la descripción de los estados.

En cuanto a los estados jerárquicos, nos encontramos los siguientes:

- Saludos: Este es el estado inicial en el que se establece la primera conexión con el usuario para darle el saludo inicial y consultar por su estado inicial.
- Negatividad: Si el estado inicial del usuario es negativo se entra en este estado para intentar animar al usuario. Se le ofrece la oportunidad que se le cuente un chiste o se le proporcione una frase motivadora para variar su estado de ánimo.
- Positividad: Si el estado inicial del usuario es positivo, se sigue con la estructura del diálogo y se le pregunta por el tiempo actual.
- Tiempo: Estado al que se llega cuando el usuario nos informa del tiempo que hace se le pregunta o por cómo es físicamente o por la estación favorita del año.

- Chiste: Estado en que se pretende subir el ánimo del usuario si este se encuentra algo negativo. Si se encuentra mejor se le pregunta por el tiempo para seguir el recorrido visto en la positividad.
- Físico: Estado para dar respuesta cuando el usuario indique su estado físico a raíz de una respuesta a una pregunta anterior. Después se le pregunta por comida o bebida, dependiendo de la respuesta.
- Música: Estado para entender interacciones musicales. Posteriormente se dirige al usuario a consultas sobre moda o mascotas.
- Película: Cuando se le consulta sobre cine o películas accede a este estado y se le dirige hacia preguntas de música o de mascotas.
- Arte: Se accede a este estado cuando indica información sobre museos o cuadros y luego se le dirige hacia consultas sobre moda.
- Moda: Accede a este estado cuando el usuario es preguntado sobre ropa y se le redirige hacia consultas sobre las acciones que tiene previsto realizar.
- Jugar: Estado relacionado con las aficiones del usuario respecto a las opciones de juego que prefiere. Tras este estado se le redirige a consultas sobre películas o arte.
- Acciones: Cuando al usuario se le realizan consultas sobre lo que va a hacer accede a este estado. Desde aquí se le presenta la oportunidad de consultar sobre otros temas o salir del sistema
- Comida: Reacciona ante preferencias alimentarias del usuario. Desde aquí se lanzan preguntas sobre películas o música.
- Bebida: Para interacciones sobre bebidas. Tras la respuesta se le lanzan preguntas sobre música o arte.
- Actividades: Cuando se pregunta al usuario sobre actividades que le gusta realizar, accede a este estado, desde el que se le dirige a consultas sobre juegos o música.
- Vacaciones: Se consulta al estado sobre sus preferencias durante las vacaciones y posteriormente se le realizan consultas sobre las actividades que le gusta realizar normalmente.
- Mascotas: Proviniendo de otros estados se llega a este respecto a preferencias sobre mascotas. Se le dirige hacia el estado de acciones visto anteriormente.
- Estaciones: Tras consultar al usuario sobre el tiempo, se le consulta la estación favorita y hace que acceda a este estado. Desde aquí se le redirige hacia los estados de vacaciones o actividades.
- Motivación: Tras el saludo inicial, si el usuario no se encuentra animado, puede acceder a este estado, si el usuario lo solicita, y se le facilita una frase

motivacional. Tras ello, si se encuentra mejor, se le consulta sobre el tiempo para encauzarlo en el flujo establecido.

- Adiós: Estado final del sistema en el que el sistema se despide del usuario y se le indica que finaliza la comunicación.

A continuación, presentamos una representación gráfica del flujo previsto de estados. Como se verá, la mayoría de los estados tienen dos salidas, ya que, como hemos indicado, algunas de las respuestas dirigen hacia un estado y otras hacia otro para dar variedad al desarrollo del sistema.

Cabe indicar que en el diagrama siguiente aparece un estado llamado “otros estados” que realmente no es un estado real del sistema, sino que cuando se le presenta al usuario la posibilidad de finalizar la comunicación o hablar de otra cosa, si prefiere seguir hablando, según lo que indique pasará a un estado concreto de los definidos y para clarificar el diagrama lo hemos indicado así ya que el estado al que acceda dependerá de lo que introduzca el usuario y puede volver a acceder a alguno de los ya visitados.

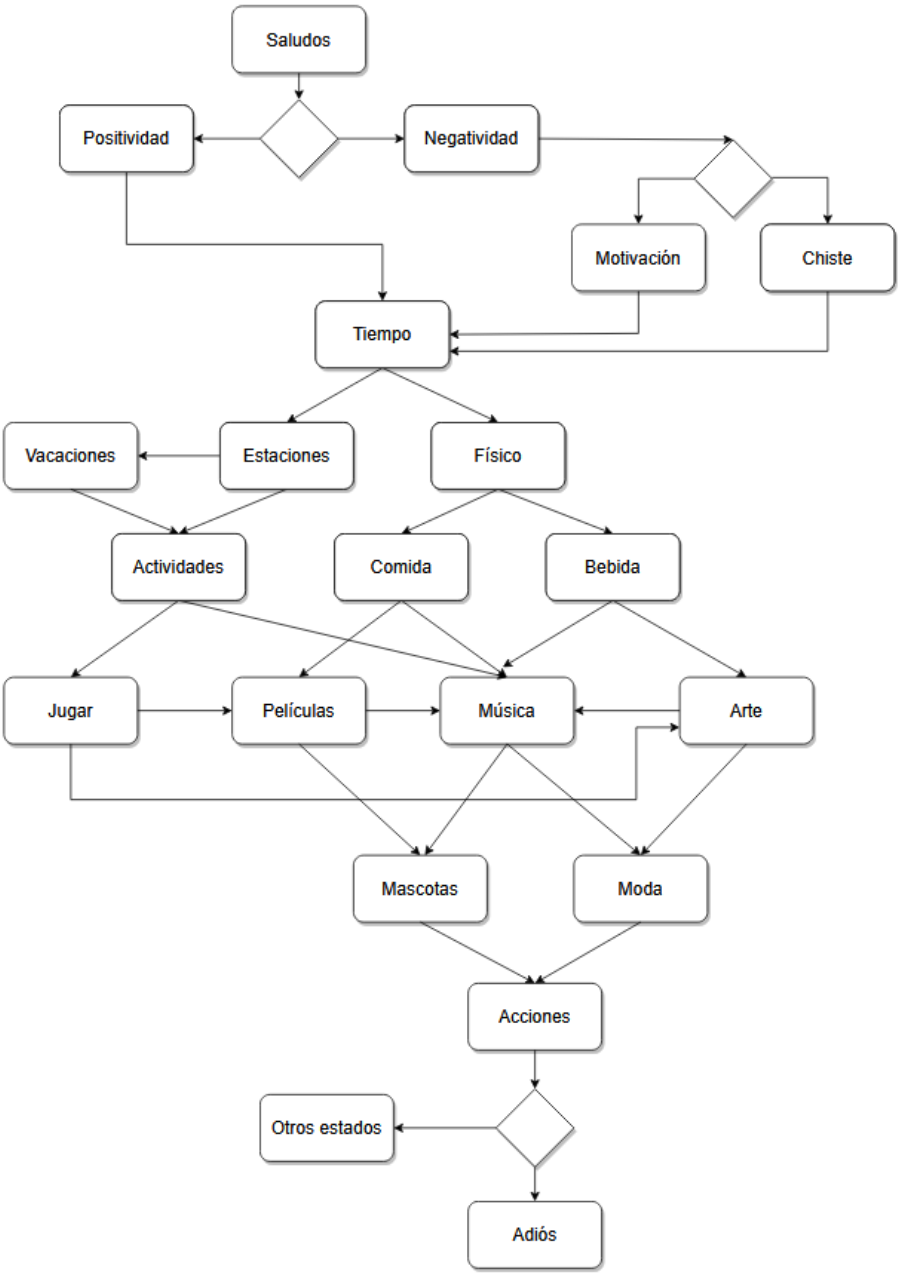


Ilustración 3 Flujograma estados jerárquicos

En cuanto a los estados libres, nos encontramos con:

- Gracias: Estado que da respuesta en caso de que el usuario dé las gracias por alguna razón
- Opciones: Estado en que se entra si el usuario consulta las posibles opciones que se pueden realizar en cualquier momento
- Nombre: Estado en el que se entra si el usuario pregunta el nombre del sistema.
- Estado: Estado al que se accede si el usuario realiza una consulta sobre el estado del sistema.

- Conexión: Aquí se accede si el usuario solicita una búsqueda de información en la red y se le informa de que el sistema se encuentra aislado y se le propone que solicite otra cosa.
- Insulto: Estado en previsión de que la reacción del usuario ante alguna situación sea la de insultar al sistema.
- Conocer: Estado similar al de conexión en el que el usuario solicite información sobre alguna temática específica y se le informa que no se dispone de ella.
- Risa: Estado de respuesta ante una reacción jocosa del usuario.
- Repetir: Estado que pretende captar si en algún momento el usuario no comprende algo que se le indica para que se repita la indicación anterior o simplemente quiere repetir la última acción, como, por ejemplo, la de contar un chiste. Se prevé que en la implantación se pueda almacenar el estado anterior y sea éste el que se ejecute.
- Edad: Estado por si el usuario quiere consultar la edad del sistema.
- Amor: Estado donde se accede si el usuario proporciona información sobre el amor.
- Salud: Estado para responder a cuestiones sobre salud.
- Médico: Estado para dar respuesta a informaciones del usuario respecto a visitas al médico.
- Familia: Estado que reacciones ante información sobre la familia del usuario

En cuanto a las respuestas que otorgará el sistema siguen la estructura que hemos presentado anteriormente y que se podrán ver más claramente en la sección de anexos esta memoria “intents.json” donde se podrá ver un listado completo de las respuestas que es capaz de entregar el sistema.

Para los estados libres, se da una respuesta de acuerdo con el estado que se encuentran y se da al usuario la libertad de seguir la conversación desde el punto que elija. Para los estados jerárquicos, además de la respuesta, se incluye una pregunta que tiene como finalidad dirigir al usuario al siguiente estado previsto en el flujograma que acabamos de ver.

Una vez tenemos toda esta estructura creada, ya se puede empezar a implementar el modelo en sí. Como ya se ha indicado, se decide dividir la implementación en dos partes bien diferenciadas: en la primera, que corresponde al archivo jackbot_model contiene toda la parte inicial de carga de los archivos correspondientes, tratamiento del archivo

de etiquetas, patrones y respuestas, la separación de los datos en conjunto de entrenamiento y test y generación de la red neuronal, con su evaluación, y guardando el modelo en un archivo específico. En la segunda parte, que corresponde al archivo jackbot_chatbot que, cargando el modelo ya generado anteriormente, define las funciones necesarias para tomar las frases que introduzca el usuario, las trabaja para dejarlas en el formato necesario y las pasa al modelo para obtener la predicción y ofrece la respuesta. También incluye la interfaz gráfica que se utilizará para interactuar con el usuario. Se podrá encontrar el código de ambos archivos incluidos en los anexos disponibles al final de la memoria. Todo el código se realiza en Python, ya que es un lenguaje de programación muy utilizado en los entornos de inteligencia artificial, y dispone de un gran volumen de librerías especializadas que simplifican la realización de muchas de las tareas.

4.2 Generación del modelo (archivo jackbot_model)

La principal librería que se va a utilizar para la interpretación del lenguaje natural utilizado por el usuario y por las interacciones utilizadas para el entrenamiento del sistema será *Spacy* [7] ya que además de disponer de muchas utilidades, también proporciona un buen soporte al idioma español. Se opta por esta librería, aunque inicialmente se realiza una prueba del sistema utilizando la librería NLTK [13] y la misma prueba con la librería *Spacy* [7], y se comprueba que en la primera se producen algunos fallos a la hora de detectar los lemas o formas canónicas de las palabras que no se producen en la segunda, además, *Spacy* [7] nos entrega un modelo establecido con multitud de procesos unificados, como la tokenización o separación en tokens o palabas y lematización o transformación en formas canónicas, y otras que aquí no se utilizarán, que en librerías como *NLTK* [12] se han de hacer por separado y eso da a un proceso más compacto y que es una gran ventaja ja que se utilizará tanto para la transformación de las frases durante el aprendizaje como durante la interacción con el usuario.

Para poder utilizarla y como primer paso, se procede a la instalación de dicha librería con el comando de Python “pip” y posteriormente se carga el modelo de lenguaje español de dicha librería en su formato grande o *large* que permitirá al sistema reconocer las palabras.

Posteriormente ya se está en disposición de realizar la importación de las librerías necesarias, como la citada Spacy [7], u otras utilizadas para la entrada y salida de datos como *json* o *pickle*, además de las librerías *numpy* y *random* de tratamiento de datos o

la librería "*re*" para poder gestionar expresiones regulares y que nos permitirá limpiar las palabras para poder trabajar con ellas. También se importarán varias librerías esenciales para el trabajo con redes neuronales que permitirán crear el modelo, las diferentes tipologías de capas y el optimizador a utilizar, y una librería específica para la separación del conjunto de datos entre entrenamiento y test con la función test_train_split.

Se procede a la carga, mediante la librería *json*, del archivo "intents.json" que se vio en el anterior apartado y donde se contienen las respuestas que se ofrecerán, las etiquetas, que identifican los diferentes estados, y los patrones que serán de donde se obtendrá el conjunto de palabras que se tratará y transformará para poder ser utilizados como base para poder entrenar el modelo de modo que pueda realizar predicciones. También se carga el modelo de vocabulario que se ha descargado anteriormente y que contiene mucha información sobre las palabras de nuestro idioma como el tipo de palabra que es y otras más, pero que en este ejercicio solamente interesará la información sobre el lema de la palabra, es decir, la forma representativa de una palabra, como, por ejemplo, de cantado, cantar.

También se generan las estructuras que se utilizarán en el sistema, como las listas donde se guardarán los lemas encontrados, las diferentes etiquetas disponibles y los pares, que almacenarán de cada frase una pareja que consistirá en los lemas de la frase y la correspondiente etiqueta. Finalmente se genera una lista con aquellas palabras conocidas como *stopwords* que corresponden a aquellas que no se quiere tener en cuenta en el estudio por ser muy comunes, como los artículos. Esta última lista se ha generado para tener un mayor control de lo que se contabiliza o no, a pesar de que ya existan listados para ello en las diferentes librerías.

Ahora ya se está en disposición de empezar a trabajar los datos. Como ya se ha indicado, se parte del archivo de intenciones o *intents* que consta de diversos bloques, cada uno de ellos con una etiqueta, un patrón y una respuesta. Precisamente de estos patrones, es de donde se generará nuestra colección de ejemplos (corpus) que servirá de entrada al modelo de predicción. Estos patrones se han de transformar a un lenguaje que entienda la red neuronal que se utilizará y como primer paso, se recorren todos y cada uno de los bloques, y en cada uno de ellos, se recorren cada uno de los patrones que contienen, que corresponderán aproximadamente a una frase. Para cada uno de estos patrones, se procede a eliminar caracteres no alfanuméricos mediante una expresión regular, además de quitar las admiraciones e interrogantes iniciales

(recordemos que la base de estos sistemas es el inglés y no contempla estos signos), se pasan las palabras a minúsculas y se eliminan los acentos.

A esta cadena, se le aplica la función “nlp()” de la librería *Spacy* [7], que genera de cada palabra o token, su lema, su función, la relación con las otras palabras de la frase y mucha otra información. Como ya se ha indicado, de toda la información de este modelo, solamente se tratarán los lemas o formas referenciales ya que permitirá detectar como una misma palabra cualquier derivación que esta. Se recorre cada uno de los tokens generados, y que no se localicen en las *stopword*s que se han definido y se almacenan. Una vez tratada una frase o patrón, se empareja con su correspondiente etiqueta y se almacenan ambas en la estructura “pares”. También por separado, se van almacenando las diferentes etiquetas que se van encontrando, pero sin realizarles ningún tratamiento.

Tanto los lemas como las etiquetas generadas se guardan en un conjunto o set para eliminar los duplicados, se pasan a listas, se ordenan y se guardan en disco ya que serán de utilidad en el archivo donde se creará el interfaz para interpretar las interacciones del usuario.

Una vez ya se dispone de los datos, se pasa a crear los datos con los que se entrenará el modelo. Para ello, partiendo del listado ordenado de lemas que se ha generado, se recorre cada uno de los pares que se han codificado y se realiza la conversión en dos pasos.

Primero se toman las frases o patrones ya lematizados, y según la lista de lemas se codifican como 0 o 1 en función de si aparecen en la lista de lemas, con lo que se obtiene una bolsa de palabras (*bag of words*) con lo que cada elemento se convertirá en una serie de 1 y 0 según las palabras del patrón correspondan con alguno de los lemas disponibles.

Por otro lado, según la lista de etiquetas que se han generado, se selecciona el segundo elemento del par que es la etiqueta y se marca con un 1 la que coincide con la etiqueta del par actual, con lo que se obtiene una codificación de *one-hot encoding* que es la que se necesita para la red neuronal y su correcto funcionamiento.

Con ello, se obtiene una nueva colección de pares, en las que el primer elemento del par es una serie de 1 y 0 según las palabras que aparecen en la sentencia inicial y el segundo elemento es una serie de 0 con un 1 en la etiqueta correspondiente. Este conjunto de pares se dividirá mediante la función *train_test_split* de la librería *sklearn* [6] en dos grupos, uno con el 80% de los registros y que será utilizada para el entrenamiento de la red neuronal y otro con el 20% restante que se utilizará para validar el

funcionamiento. Además, estos conjuntos se dividirán en dos partes x e y donde en la primera parte se pone la parte correspondiente a los patrones codificados y que corresponden a la entrada de la red neuronal y en la segunda parte, la y, se almacenan las salidas correspondientes.

Se realiza ahora el modelo de la red neuronal utilizada con varias capas ocultas y con una capa de entrada del tamaño de los datos que se han almacenado como primer elemento de los datos de entrenamiento y que corresponderá al número de lemas únicos encontrados. La capa de salida tendrá el tamaño de las salidas esperadas que coincidirá con el número de etiquetas únicas generadas. A esta red se le añaden varias capas de *dropout* que ayudarán a evitar el sobreajuste y que el sistema se especialice demasiado en los datos de entrenamiento y no sea capaz de realizar una nueva predicción mediante el bloqueo de diversas neuronas para que no aprendan y evitar un sobreajuste.

A esta red se le aplicará un optimizador, utilizado para la mejora del rendimiento de aprendizaje de la red, inicialmente basado en el SGD o Gradiente descendiente estocástico pero que tras varias pruebas se decide cambiarlo por el optimizador *Adam* como veremos más adelante en el apartado de ajustes, que tiene como objetivo minimizar la función de error del sistema y obtener de la forma eficiente el mejor mínimo posible y de ese modo mejorar las predicciones que pueda realizar y poder alcanzar un bien nivel de exactitud (*accuracy*) que será la medida sobre la que evaluaremos el sistema.

Solamente se realiza el cálculo del *accuracy* y no se exploran otras medidas ya que solamente se utilizará como comparativa en las diferentes pruebas, ya que no se busca un nivel de predicción muy alto, sino que las previsiones realizadas, aunque no correspondan a lo que se preveía de ellas, generen resultados igualmente coherentes y que no causen respuestas del sistema que dificulten la conversación.

El modelo se compila y se entrena por primera vez y se guarda en disco para su uso por otro archivo, que será donde se implementará la interfaz de interacción con el usuario. Finalmente se incluyen dos líneas para el cálculo del rendimiento del sistema tanto para datos nuevos o de prueba, como para datos con los que se ha entrenado el sistema o de entrenamiento y que servirán en futuras fases para poder mejorar los resultados obtenidos. Posteriormente y una vez implementado completamente el modelo, se realizarán diversas pruebas y ajustes para mejorar el rendimiento total del proyecto y que veremos más adelante.

4.3 Generación del interfaz (archivo jackbot_Chatbot)

Se inicia el proceso con la importación de las librerías necesarias para la carga y lectura de archivos, además de las necesarias para el tratamiento de la información, de modo similar al realizado en el anterior archivo. Posteriormente se procede a la carga de los archivos necesarios que serán el archivo de intenciones de donde se elegirán las respuestas a entregar al usuario, los archivos de lemas y etiquetas generados durante la creación del modelo, el propio modelo que se guardó tras el entrenamiento, el modelo de vocabulario de *Spacy* [7] ya utilizado en la creación del modelo, y las *stopwords* que se definieron también en el anterior archivo.

Como primer paso, se definen dos funciones *lemmatize_input* y *bow_f* que servirán para imitar el proceso que se realizó con los datos iniciales para procesar las entradas del usuario, en ellas, primero se realiza la limpieza de los datos de entrada introducidas, se eliminan puntuaciones, acentos, se pasan a minúsculas y se transforman en sus correspondientes lemas para posteriormente generar una bolsa de palabras que será la que se compararán con las que presenta el sistema para producir la predicción gracias al modelo implantado.

Posteriormente, se implementan otras tres funciones *answer*, *select_answer* y *predict_label*. La primera de ellas, toma la sentencia que ha introducido el usuario y la pasa a la función *predict_label* que utilizando las funciones del punto anterior las transformarán a un tipo de dato utilizable por el modelo, realiza una predicción de las posibles opciones y las entrega ordenadas de mayor a menor probabilidad y las pasa a la función *select_answer* donde escogerá la de mayor probabilidad, y de esa etiqueta, escogerá aleatoriamente una respuesta de entre las posibles que hay en el archivo de intenciones y la devuelve al usuario. En la función *answer* también se realiza una comprobación para ver si la etiqueta devuelta es la que corresponde al estado repetir y devuelve el estado anterior, o si detecta que no se ha localizado ninguna respuesta por parte del modelo, solicita al usuario que detalle su interacción de otro modo.

Finalmente, mediante la librería *thinker* se generan los diferentes elementos que formarán parte de la interfaz del usuario como el cuadro de texto donde aparecerán los mensajes, el mensaje inicial, el cuadro de texto donde el usuario introducirá sus sentencias y la propia ventana del chat. Utilizando la función *mainloop* de la librería citada se genera un bucle infinito para la ejecución de la ventana generada y el inicio del funcionamiento del bot generado.

4.4 Realización de ajustes para mejorar la fiabilidad

Se realizan varias rondas de implementación de ajustes y pruebas del sistema, para detectar los diferentes puntos de mejora e intentar obtener un resultado aceptable.

Una vez se dispone del modelo implementado y con el primer entrenamiento realizado, se revisan los valores de las validaciones que se han obtenido tras el primer entrenamiento y se ve que los porcentajes de acierto se encuentran alrededor de un 30% quedando muy por debajo de un porcentaje aceptable para el resultado buscado.

Como primer paso, se decide la inclusión de nuevos patrones que pueda reconocer el modelo para poder ampliar un poco el conjunto de entrenamiento. Este conjunto no es muy grande, ya que se parte de un volumen de patrones bajo, lo que afectará de todos modos al rendimiento del sistema y su capacidad de predicción, pero como se ha decidido la construcción del conjunto de datos manualmente para un mayor control, no se baraja la opción de incluir conjuntos masivos de datos más genéricos.

Con la inclusión de algunos patrones y tras el correspondiente entrenamiento mejora un poco el porcentaje de éxito, pero sin llegar a un porcentaje aceptable por lo que se buscan otros posibles ajustes que realizar.

El siguiente ajuste que realizar se centra en el optimizador de la red neuronal que se había elegido en primer lugar. Para ello, se realizan pruebas con varios optimizadores, que calcularán los mínimos de las funciones de coste, y se comprueba que el que ofrece mejores resultados que el actual es el optimizador *Adam*, como hemos comentado anteriormente, por lo que se utiliza este optimizador en lugar del que habíamos seleccionado. Solamente con este cambio se obtiene una mejora de la capacidad de predicción del sistema obteniendo un porcentaje de *accuracy* del 45%.

Una vez llegados a este punto y con la realización de varias pruebas con diferentes usuarios, se decide retomar una segunda ronda de ajustes que dividiremos en dos partes que vamos a describir.

Como primera parte se decide incidir en los datos iniciales. Se analiza la totalidad de las interacciones erróneas que se han producido durante las pruebas, se toman aquellas que se considera más significativas para el resultado final y se decide etiquetarlas e incluirlas en el listado de patrones que se tenía anteriormente. También se decide no modificar las etiquetas de las que se disponía inicialmente, al considerarse que el flujo inicialmente planteado ya funciona siempre y cuando se detecten correctamente estas

etiquetas. Se realizan diversos entrenamientos posteriores pero el rendimiento no mejora especialmente.

También se observa que modificando la semilla a partir de la que se deciden los registros para realizar la partición también influyen en el resultado final, se realiza un bucle de prueba para probar todos los posibles valores y detectar el óptimo. Con el valor que ha dado una mejor valoración será con el que se realizará finalmente la partición y será la semilla que se definirá para que siempre que se realice la partición se realice del mismo modo.

Como segunda parte de los ajustes que se realizan, se decide incidir sobre los hiperparámetros de la red neuronal, que son todos aquellos parámetros que es posible modificar de cara a una optimización del resultado final obtenido. Para ello, se realizan bucles realizando el mismo proceso, pero modificando cada vez alguno de los hiperparámetros implicados para ver con cuales de ellos se obtiene un mejor resultado. Algunas de las pruebas que se realizan implican el *learning rate* o el ratio de aprendizaje que implica el ratio que se tendrá en cuenta de cara a la reducción de la función de error y que ayudará a mejorar la capacidad de aprendizaje, se varían la cantidad de neuronas, el porcentaje de *dropout* entre las capas que se refiere al porcentaje de neuronas que se desconectarán para que no aprendan y evitar el sobreajuste, así como el *batch_size* o tamaño de registros que se tienen en cada una de las pruebas o las épocas que son el número de repeticiones que se realiza de todo el conjunto de datos, que se ha realizado en los subgrupos que se han definido en el tamaño de registros.

Con todas estas pruebas, se llega finalmente a una *accuracy* o porcentaje de datos predichos correctamente respecto el total de datos predichos del 58.89% y que se decide dejar como resultado final del modelo ya que no se logran mejorías relevantes de este dato.

5 Resultados

Como resultado final una vez seguida la metodología marcada en el punto anterior, se obtiene un modelo que mediante una interfaz visual permite al usuario mantener una conversación parcialmente guiada con el sistema, siguiendo una sucesión de estados, pero que también permite dar respuesta a otros temas que son de interés para el usuario, pero que inicialmente no se haya considerado incluirlos en le flujograma previsto. Con esto se pretendía generar una sensación de acompañamiento con el usuario que, al menos en parte, se ha conseguido lograr en base a las reacciones de los usuarios con los que se han realizado las pruebas.

Durante las pruebas que se han ido realizando, se ha realizado un doble análisis de lo que se iba obteniendo. Por un lado, se ha analizado la capacidad de predicción del sistema frente a las interacciones del usuario, comprobando si la intención prevista coincidía con la real, en el caso de los datos del conjunto de validación que venían previamente etiquetados y en el caso de las pruebas con usuarios, con la intención que se podía extraer de la frase o interacción que se ha introducido. En este análisis se ve que, si bien en muchas ocasiones no era exactamente la misma intención prevista con la real, en muchas ocasiones si podían ser equivalentes o no excesivamente discordantes, como, por ejemplo, que se hable sobre comprar unos pantalones y se haya previsto como una actividad y se interprete como moda, o casos similares. Esto hace que el segundo análisis que se realizaba simultáneamente al primero y que consistía en ver si se producía un flujo correcto en las interacciones que permitieran mantener una conversación y responder al usuario, diera un resultado más satisfactorio de cara a las sensaciones experimentadas, ya que nos indicaban que, si bien, en alguna ocasión el resultado no era el esperado, permitía seguir con la conversación y proporcionar cierta sensación de no estar solo y de acompañamiento.

Pero dado que este acompañamiento no es una entidad medible y que podamos calcular, se ha decidido medir el baremo de fiabilidad del sistema mediante la medida en la que es capaz de ajustar las sentencias introducidas por el usuario a la intención que tiene, es decir, si la respuesta que ofrece es coherente con la interacción que ha realizado el usuario. Solamente se tiene en cuenta esta medida de fiabilidad y no se analizan otras diferentes ya que no se busca un modelo que obtenga unos altos niveles de predicciones acertadas si no que se encuentren dentro de las posibilidades que cabría esperar y que no ocasionen resultados excesivamente estridentes para el usuario.

A pesar de que mientras se observan las interacciones de los usuarios no se detecta un índice de fallos exagerado respecto al de aciertos, cuando se calculan mediante el conjunto de validación, tal y como se ha comentado anteriormente, se obtiene un índice de aciertos sobre el total de interacciones, lo que se conoce como *accuracy* del 58.89% que aunque no es excelente, sí que es suficiente, desde un punto de vista subjetivo, para proporcionar cierta sensación de estar conversando con alguien y dar esa sensación de acompañamiento que se estaba buscando.

Podemos ver una imagen del resultado final que observa el usuario:

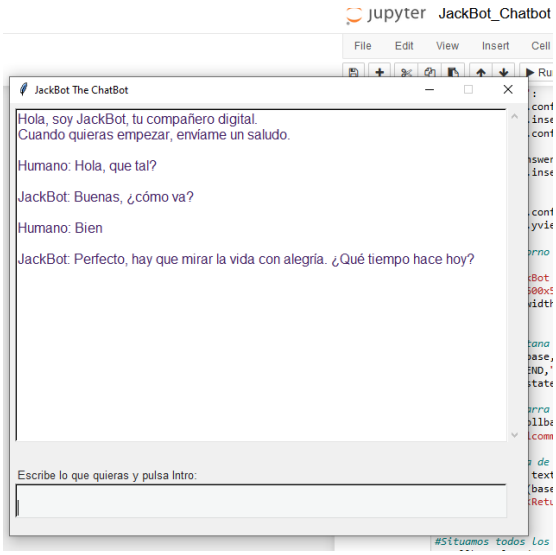


Ilustración 4. Funcionamiento del chatbot

6 Discusión

Una vez finalizado el desarrollo del sistema, se ha comprobado que se aproximan los resultados que se han logrado respecto a los objetivos marcados inicialmente ya que ha permitido aplicar los conocimientos adquiridos a lo largo del grado en la creación de una herramienta que permita a cualquier usuario mantener una conversación con una entidad que permita generar una sensación de acompañamiento y que alivie la sensación de soledad, tan presente en nuestra sociedad, dándole la oportunidad de hablar con alguien en momentos en los que no tiene a nadie con quien hacerlo.

Se ha partido de un archivo base de intenciones y etiquetas que tienen el objetivo de entrenar un modelo de aprendizaje que permita reconocer las interacciones que realice el usuario y dar una respuesta a éstas. Si bien se ha comprobado que se hubiera requerido un volumen mucho mayor de este conjunto de datos para hacer frente a situaciones concretas, cumple su objetivo de marcar un flujo conversacional y establecer una conexión con el usuario. El gran problema que plantea un idioma como el español es la gran cantidad de frases hechas o dobles sentidos que pueden tener las frases que se introducen y que con un volumen de datos como el que se ha generado se queda muy lejos de poder cubrir muchas de las posibles opciones, sí que da una base para las situaciones más habituales.

La elección de esta estructura basada en este archivo de intenciones también da la posibilidad de una gran flexibilidad a la hora de su aplicación a otros entornos, ya que se ha comprobado que una modificación en las preguntas que se incluyen en las respuestas puede hacer que se altere el flujo establecido abriendo nuevas posibilidades de desarrollo. Esta gran ventaja también implica una gran limitación por el hecho de que impone una rigidez frente al amplio abanico de posibilidades que se pueden producir en las interacciones del usuario según su educación, nivel cultural o intereses, pero que sí que cubren las necesidades básicas de una conversación común y sin grandes pretensiones.

Por ello, se ha ajustado el desarrollo a esta faceta básica, dado que la intención de intentar abarcar un amplio abanico de posibilidades hubiera requerido un archivo de intenciones mucho más extenso, tanto a nivel de posibles etiquetas o estados, como de patrones de referencia para cubrir todas las temáticas posibles, lo que habría resultado una tarea imposible de abarcar, especialmente con un plazo temporal limitado como el disponible para el presente trabajo de fin de grado.

7 Conclusiones

7.1 Conclusiones

En el presente trabajo se ha estado hablando mucho del término acompañamiento y que es el objetivo que se pretendía alcanzar. Este término en su sentido más amplio es muy abstracto y difícil de alcanzar, ya que se trata de una sensación y como tal es totalmente subjetiva y cada individuo tiene su forma de entenderlo. Para ello, se ha optado por una visión del acompañamiento como aquella sensación que simplemente te aporte la posibilidad de mantener una conversación con alguien para aquellos momentos de soledad en que no se puede hacer con nadie más y al que se le pueda explicar sus inquietudes sin más pretensiones. Por ello, se ha enfocado este trabajo a un nivel básico de compañía sin grandes fuentes de información o sin grandes habilidades sino enfocado simplemente a proporcionar la posibilidad de explicar lo que ha sucedido durante el día y que te responda, sobre todo enfocado a aquellas personas que no disponen de la posibilidad de hacerlo con nadie más.

Precisamente la imposición de este objetivo de trabajo ha obligado a limitar las opciones disponibles y el flujo establecido, ya que se ha comprobado en las pruebas realizadas, según la persona que las realizaba, mostraba unos intereses u otros y por ello se ha optado por establecer una base común y trabajar sobre la misma.

Por esta misma razón, no se ha buscado un alto grado de aciertos en las predicciones realizadas en referencia a las interacciones con el usuario, sino que estos posibles fallos no fueran demasiado chocantes para el usuario y no se perdiera el flujo mantenido hasta el momento. Además, para lograr un rendimiento de predicción más alto hubiera requerido un conjunto de datos de entrenamiento mucho mayor y más diversificado, pero que, como se ha indicado anteriormente, se ha optado por limitarlo para obtener un comportamiento más básico, pero también mucho más genérico y adaptado al público general.

Finalmente, no ha sido posible la implementación del sistema en una aplicación final que fuera portable a diferentes entornos o ejecutable desde otras plataformas debido a la limitación temporal, pero este objetivo secundario se deja como una posible opción dentro de las posibles líneas de futuro.

7.2 Líneas de futuro

En cuanto a las posibilidades que nos presenta este sistema de cara a un futuro pueden ser diversas y enfocadas a distintos puntos del sistema tanto para dotarlo de nuevas capacidades, para hacerlo más accesible a los usuarios o ambas.

Una de las posibles líneas de futuro a seguir se podría centrar en la ampliación y modificación del archivo de intenciones. Como ya se ha indicado, este archivo es la base de todo el sistema y el que permitiría una mayor adaptabilidad. Una opción es aumentar el número de patrones en cada una de las etiquetas o incluso aumentar el número de etiquetas para aumentar el abanico de posibles estados y el flujograma asociado y aportar de un mayor vocabulario y mejorar la capacidad de predicción del sistema. Una ampliación del conjunto de entrada permitiría la creación de un modelo entrenado más eficientemente y que se podría enfrentar mejor a todas las posibilidades. Otra opción sería la de especializar el sistema enfocando los estados y las respuestas, y, por tanto, el flujograma a situaciones concretas, como un asistente en una tarea concreta o dirigido a un entorno más específico con un vocabulario o argot más restrictivo.

Otra posibilidad podría ser la de dotar al sistema de memoria, en el sentido que diera la oportunidad de guardar las conversaciones que se mantengan y permitir su etiquetado para que formen parte del conjunto de entrenamiento disponible y que permitan entrenar continuamente al sistema para mejorar su rendimiento y efectividad.

Finalmente, también se puede optar por implementar el sistema en una aplicación que se pueda ejecutar en cualquier entorno o incluso en dispositivos móviles o también incluirla en plataformas de sistemas de inteligencia artificial como *Hugging Face* [14]que permitan a cualquier usuario acceder al sistema y probarlo libremente sin la necesidad de ejecutar el sistema en una plataforma como *Jupiter Notebook* [8].

7.3 Seguimiento de la planificación

La planificación realizada en un primer momento se estableció desde un punto de vista realista y que proporcionara un mayor volumen de tiempo a la búsqueda de información y al desarrollo del archivo de intenciones, la base del sistema, y que ha permitido que en todo momento, como se ha podido observar en los informes de seguimiento realizados a lo largo de la asignatura, se pudiera seguir la planificación prevista sin tener

que hacer ajustes posteriores o modificar o anular ninguna de las tareas que se habían planificado.

También se ha podido realizar el desarrollo previsto con la metodología establecida inicialmente y que ha permitido la obtención del sistema que se ha presentado con la funcionalidad establecida en un primer momento y cumpliendo los objetivos principales marcados. Como se ha comentado, sí que se podría haber dedicado más tiempo a ampliar el desarrollo del archivo de intenciones usado para el entrenamiento del sistema de predicción, pero eso hubiera afectado la planificación establecida y, como se ha indicado, con el desarrollo actual se ha cumplido el objetivo inicial establecido por lo que no se ha considerado ampliar el tiempo asignado a esta tarea y que pudiera impactar en la capacidad de entrega de un modelo final funcional.

8 Glosario

A continuación, indicamos aquellos términos o acrónimos utilizados a lo largo de la memoria:

- **Chatbot**: Se trata de un asistente que se comunica con el usuario a través de mensajes, habitualmente de texto, y que, en general, se enfocan a tareas específicas
- **Inteligencia Artificial**: Se basa en la creación de algoritmos que permitan a los sistemas informáticos realizar tareas de toma de decisiones basadas en el comportamiento humano.
- **NLP**: Procesamiento del lenguaje natural. Conjunto de herramientas que permiten interpretar el lenguaje natural se las personas por las máquinas.
- **Python**: Lenguaje de programación con muchas aplicaciones en la inteligencia artificial, además de muchas otras.
- **Librerías**: conjunto de aplicaciones de un lenguaje de programación que proporcionan acceso a tareas específicas, proporcionando reaprovechamiento de las tareas.
- **Sklearn y Spacy**: Librerías del lenguaje de programación Python especializadas en tareas de Inteligencia Artificial.
- **Flujo de estados**: En un chatbot, listado de posibles estados o situaciones en las que se puede encontrar un sistema que provoquen una respuesta de éste.
- **Intenciones**: Dentro del NLP es una interpretación del significado de una interacción del usuario con el sistema y que puede provocar un cambio de estado del sistema y ofrecer una respuesta determinada.
- **Jupiter Notebook**: Plataforma para el desarrollo de programas en lenguajes como Python similar a la plataforma Google Colab, pero utilizada a través del aplicativo Anaconda.
- **Tokenización:** Dentro del NLP es el proceso que permite dividir un texto en partes, que usualmente pueden ser palabras u otras partes de la oración de cara a su análisis de cada una de esas partes. Aquellos que no correspondan a palabras se han de tratar eliminando puntuaciones u otros caracteres.
- **Lematización:** Consiste en una vez se dispone de las palabras claramente diferenciadas, una de las opciones para su tratamiento es extraer el lema de cada una de ellas, es decir, la raíz de cada una de las palabras, de modo que

simplifica las posibles apariciones de una palabra en sus diferentes formas a una sola, por ejemplo, un verbo a su infinitivo.

- **Corpus:** Conjunto de textos o audios captados del lenguaje natural que constituyen un set de datos y que tienen un procesado posterior que usualmente se llama notación con el etiquetado de su vocabulario y otras características como la tipología de las palabras, su lema u otro tipo de información.
- **JSON:** Se trata de un formato sencillo de intercambio de datos basados en la notación de objeto de Javascript y se utiliza en múltiples aplicaciones como formato de intercambio de información textual.
- **One-hot encoding:** Técnica que consiste en codificar una columna creando una nueva columna para cada uno de los posibles valores de la columna inicial codificando con un 1 en aquella en que corresponde el valor con el que se ha de codificar.
- **Bag of words(bolsa de palabras):** Utilizado en las técnicas de lenguaje natural consiste en representar un documento que contiene las palabras que aparecen en el documento pero sin importar el orden, solamente importa si aparecen.
- **Dropout:** Técnica que reduce el sobreajuste en las redes neuronales y que omite aleatoriamente neuronas durante el entrenamiento de la red neuronal.
- **Accuracy:** Computa el porcentaje de etiquetas predichas que corresponden con las reales, es decir, aquellas que el sistema predice correctamente.

9 Bibliografía

Bibliografía utililzada:

- [1] *Productos y servicios/Publicaciones/Colección de cifras INE 2020* [en línea] [consulta: 30 de septiembre de 2022]. Disponible en: https://www.ine.es/ss/Satellite?L=es_ES&c=INECifrasINE_C&cid=1259952645332&p=1254735116567&pagename=ProductosYServicios%2FINECifrasINE_C%2FPYSDetalleCifrasINE
- [2] *Contra la soledad "que duele"* [en línea] [consulta: 30 de septiembre de 2022]. Disponible en: https://www2.cruzroja.es/web/ahora/-/contra-soledad-que-duele
- [3] *¿Cómo funciona Alexa?* [en línea] [consulta: 02 de octubre de 2022]. Disponible en: https://computerhoy.com/reportajes/tecnologia/como-funciona-alexa-1037243
- [4] *Bigscience/Bloom* [en línea] [consulta: 06 de octubre de 2022]. Disponible en: https://huggingface.co/bigscience/bloom
- [5] Rasa [en línea] [consulta: 05 de noviembre de 2022]. Disponible en: https://rasa.com/
- [6] *Sklearn* [en línea] [consulta: 05 de noviembre de 2022]. Disponible en: https://scikit-learn.org/
- [7] *Spacy* [en línea] [consulta: 02 de noviembre de 2022]. Disponible en: https://spacy.io/
- [8] *Jupiter Notebook* [en línea] [consulta: 02 de noviembre de 2022]. Disponible en: https://jupyter.org/
- [9] *OpenAI* [en línea] [consulta: 05 de noviembre de 2022]. Disponible en: https://openai.com
- [10] *Estudi i creació d'un assistent virtual(xatbot)* [en linea] [consulta: 06 de octubre de 2022]. Disponible en: http://hdl.handle.net/10609/71725
- [11] Tfg_Noelia_MARTINEZ_DIAZ [en línea] [consulta: 06 de octubre de 2022]. Disponible en: https://oa.upm.es/68080/1/TFG_NOELIA_MARTINEZ_DIAZ.pdf
- [12] *JSON* [en línea] [consulta: 05 de noviembre de 2022]. Disponible en: https://www.json.org/json-es.html
- [13] NLTK [en línea] [consulta: 05 de noviembre de 2022]. Disponible en: https://www.nltk.org
- [14] Hugging Face [en línea] [consulta: 02 de noviembre de 2022]. Disponible en : https://huggingface.co/

Otra bibliografía para documentación:

- *Tutorial: Construye tu primer chatbot usando NLTK y Keras* [en línea] [consulta: 04 de octubre de 2022]. Disponible en: https://planetachatbot.com/tutorial-chatbot-usando-nltk-keras/
- *Storytelling And Bot Making* [en línea] [consulta: 25 de septiembre de 2022]. Disponible en: https://chatbotsmagazine.com/storytelling-and-bot-making-5fb1b5eaff9
- *Storytelling with Chatbots* [en línea] [consulta: 25 de septiembre de 2022]. Disponible en: https://medium.com/@danpalan1/storytelling-with-chatbots-70a96eaef0ad
- *Generative chatbots using the seq2seq model!* [en línea] [consulta: 06 de octubre de 2022]. Disponible en: https://towardsdatascience.com/generative-chatbots-using-the-seq2seq-model-d411c8738ab5
- *Keras* [consulta: 04 de noviembre de 2022]. Disponible en: https://keras.io/

10 Anexos

10.1 Anexo 1. Muestra de una intención del archivo de intenciones intents.json

El archivo completo se encuentra en: https://github.com/KikoDM/UOC/tree/main/TFG


```
{"intents": [
      {"etiqueta": "saludos",
       "patrones": ["Hola, ¿qué tal?",
                    "Buenos días",
                    "Buenas tardes",
                    "Buenas noches",
                    "Hola",
                    "Aquí estamos",
                    "He vuelto",
                    "Buenas, ¿Cómo va todo?",
                    "Ei, cuanto tiempo sin verte",
                    "Hola, un día más por aquí",
                    "Encantado de verte por aquí",
                    "Hey, ¿qué tal?",
                    "Hola, ¿cómo va todo?",
                    "Buenas",
                    "Saludos",
                    "Ei, ¿Qué tal?"],
       "respuestas": ["Hola, un día más por aquí, ¿qué tal todo?",
                      "Encantado de verte por aquí, ¿cómo te encuentras?",
                      "Hola, ¿cómo va todo?",
                     "Buenas, ¿cómo va?",
                      "Hey, ¿qué tal?"]
      },

}
```

10.2 Anexo 2. Generación del modelo predictivo (archivo jackBot_model.ipynb)

El archivo completo se encuentra en: https://github.com/KikoDM/UOC/tree/main/TFG

Mostramos aquí la parte del modelo que se encargará del tratamiento inicial del conjunto de datos antes de la codificación de los mismos:

```
#Generamos las estructuras que vamos a utilizar a lo largo del modelo
words=[]
lemmas = []
etiquetas = []
pares = []
stops = ['a','yo','tu','el', 'nosotros', 'vosotros', 'ellos', 'me', 'mi', 'te', 'ti', 'la', 'los', 'las', 'y', 'o', 'al','del','ya','que']
#Leemos los datos que hemos guardado del archivo de intenciones, y para cada uno de los patrones o frases
#Primero eliminamos carácteres no deseados, pasamos a minúsculas y quitamos acentos
#Aplicamos el método nlp de spacy para poder obtener los lemas de cada una de las palabras del patrón
#Almacenamos en pares el conjunto de lemas de cada patron con su correspondiente etiqueta
for intent in intents['intents']:
  for patron in intent['patrones']:
    w=re.sub(r'[¿¡]', "", patron)
    w=re.sub(r'\W+', " ", w)
    w=w.lower()
    w=re.sub('á', 'a', w)
    w=re.sub('é', 'e', w)
    w=re.sub('í', 'i', w)
    w=re.sub('ó', 'o', w)
    w=re.sub(r'[üú]', 'u', w)
    doc = nlp(w)
    words=[]
    for token in doc:
```

```
        if token.text not in stops:
            words.append(token.lemma_)
            lemmas.append(token.lemma_)
    pares.append((words,intent['etiqueta']))
    if intent['etiqueta'] not in etiquetas:
        etiquetas.append(intent['etiqueta'])

#Creamos sets de los lemmas y las etiquetas para tener datos únicos ordenados y los guardamos para usuarlos en el bot
lemmas=sorted(list(set(lemmas)))
etiquetas=sorted(list(set(etiquetas)))
pickle.dump(lemmas,open('lemmas.pkl','wb'))
pickle.dump(etiquetas,open('etiquetas.pkl','wb'))
```

10.3 Anexo 3.Creación interfaz de interacción (archivo jackBot_chatBot.ipynb)

El archivo completo se encuentra en: https://github.com/KikoDM/UOC/tree/main/TFG

Mostramos aquí la parte del modelo que se encargará de la interfaz gráfica con la que interactuará el usuario

```
#Creamos la ventana de interacción con el usuario
import tkinter
from tkinter import *
#Definimos la interacción
def send(event):
    msg = EntryBox.get("1.0",'end-1c').strip()
    EntryBox.delete("0.0",END)
    if msg != '':
        ChatLog.config(state=NORMAL)
        ChatLog.insert(END, "Humano: " + msg + '\n\n')
        ChatLog.config(foreground="#442265", font=("Arial", 12 ))
        res = answer(msg)
        ChatLog.insert(END, "JackBot: "+ res + '\n\n')
        ChatLog.config(state=DISABLED)
        ChatLog.yview(END)
#Creamos el entorno
base = Tk()
base.title("JackBot The ChatBot")
base.geometry("600x500")
base.resizable(width=FALSE, height=FALSE)

#Creamos la ventana de Chat con un mensaje de inicio
ChatLog = Text(base, bd=2, bg="white", height="8", width="50", font="Arial",)
ChatLog.insert(END,"Hola, soy JackBot, tu compañero digital.\nCuando quieras empezar, envíame un saludo.\n\n")
ChatLog.config(state=DISABLED)
```

```
#Añadimos una barra deslizante
scrollbar = Scrollbar(base, command=ChatLog.yview, cursor="heart")
ChatLog['yscrollcommand'] = scrollbar.set

#Creamos la caja de entrada de texto con un mensaje informativo
lbl=Label(base, text="Escribe lo que quieras y pulsa Intro:", font=("Arial", 10),fg="#262626")
EntryBox = Text(base, bd=2, bg="white",width="29", height="5", font="Arial",
background="#f5f7f7")
EntryBox.bind('<Return>', send)

#Situamos todos los objetos en su lugar
scrollbar.place(x=576,y=6, height=386)
ChatLog.place(x=6,y=6, height=386, width=570)
EntryBox.place(x=6, y=440, height=40, width=570)
lbl.place(x=6,y=420,height=20, width=215)

#Arrancamos el bucle que ejecutará el bot
base.mainloop()
```