#HPECDSTechChallenge

# HEALTHY AI, TU ASISTENTE EN PRIMEROS AUXILIOS

La gran solución al problema planteado por

CDS | Hewlett Packard Enterprise

Del grupo **Cebollitas** compuesto por

Alejandra Inés Lapieza Pérez, Raúl Sanz Mancebo y Junjie Wu

Estudiantes de 3º en Ingeniería Matemática Aplicada al Análisis de Datos (Data Science) de la Universidad Europea de Madrid

ABRIL 2024

# ÍNDICE DE CONTENIDO

# ÍNDICE DE TABLAS



# ÍNDICE DE DIAGRAMAS

# INTRODUCCIÓN

La atención médica eficiente es fundamental para salvar vidas en situaciones de emergencia. Sin embargo, **muchas personas carecen de conocimientos adecuados sobre primeros auxilios**, lo que puede llevar a retrasos en la asistencia o incluso a acciones incorrectas en momentos críticos. Para abordar este desafío, **Healthy AI** planteó la necesidad de desarrollar una solución innovadora que pudiera **proporcionar orientación rápida y precisa en situaciones de emergencia.**

La tarea consistía en crear una **herramienta que aprovechara la inteligencia artificial** para ofrecer instrucciones claras y pertinentes a los usuarios en momentos de crisis.

> Es fundamental destacar que esta solución no pretendía reemplazar a los profesionales médicos ni sustituir la atención médica cualificada.

La solución propuesta para este desafío es el **desarrollo de un ChatBot** integrado con la plataforma de mensajería **Telegram**. Este ChatBot permitiría a los usuarios enviar mensajes de texto o grabaciones de audio describiendo la emergencia que están enfrentando. A partir de esta entrada, **la inteligencia artificial detrás del ChatBot analizaría la situación y proporcionaría instrucciones** paso a paso sobre las acciones que el usuario debe tomar para brindar ayuda inicial de manera efectiva.

> Cabe destacar que esta interfaz puede ser sustituida por otra en caso de ser necesario

En esta documentación, se detallará el proceso de desarrollo de la solución, incluyendo la arquitectura del sistema y los algoritmos de inteligencia artificial utilizados.

# DATOS

## PRIMEROS AUXILIOS

Para obtener **información fiable** sobre los procedimientos a seguir en casos de primeros auxilios, **se seleccionó la Universidad de Valencia como fuente primaria**. Esta institución cuenta con una página web que presenta una **variedad de casos de primeros auxilios**, cada uno de los cuales está descrito con detalle, **incluyendo los procedimientos recomendados.**

## DATOS PARA EL MODELO

**Hemos seleccionado varios tipos de casos de primeros auxilios**: quemadura química, intoxicación o quemadura por inhalación, quemadura severa, intoxicación alimenticia, picaduras, fuego, hemorragia o heridas, electrocución, atragantamiento, accidente cardiopulmonar, ataque epiléptico y ataque de hipoglucemia.

Para cada caso de primeros auxilios, se realizó un **proceso de identificación de aproximadamente 8 palabras clave relacionadas**. Este proceso se llevó a cabo mediante una sesión de "brainstorming" inicial, seguida de una selección de palabras que no fueran ambiguas y que tuvieran pocos sinónimos, garantizando así la precisión y la coherencia en la información proporcionada.

| CATEGORY | KEYWORDS |
|---|---|
| **CHEMICAL BURN** | acid, red, swollen, reaction, contact, product, spilled, cleaning |
| **BURN INTOXICATION INHALATION** | cough, breathe, steam, smoke, gas, air, lung, throat |
| **SEVERE BURN** | blister, oil, fire, heat, fur, red, boil, hot |
| **FOOD POISONING** | diarrhea, vomit, stomach, nausea, meal, eat, fever, abdominal |
| **STING** | insect, snake, bee, fur, swollen, poison, wasp, spider, inflammation |
| **FIRE** | forest, firefighter, extinguisher, smoke, exit, hose, alarm, suffocation, breathe |
| **HEMORRHAGE** | blood, court, wound, pressure, deep, cold, weak, cut, loss |
| **ELECTROCUTION** | plug, voltage, electric, shock, current, strain, wire, contact, download |
| **CHOKING** | swallow, meal, stuck, suffocation, drown, throat, obstruction, food, obstruction |
| **CARDIOPULMONARY** | heart, breathe, air, defibrillator, chest, revive, pulse, cardiac, unconscious |
| **EPILEPSY** | epileptic, muscular, rigid, seizure, spasm, drop, tremble, shaking, seizure |
| **HYPOGLYCEMIA** | pale, tremble, sweat, hunger, nausea, fatigue, anxiety, sugar, insulin |

**Tabla 1.** Emergencias Médicas y Síntomas Relacionados

## DATOS DE PRUEBA

Con el objetivo de crear un **conjunto de datos de prueba para el modelo**, se solicitó a ChatGPT4 que generara una serie de textos **simulando llamadas de emergencia** para cada tipo de situación de emergencia. De esta manera, se obtuvo un conjunto de datos clasificados que permitió realizar pruebas de aprendizaje supervisado y validar la eficacia del modelo en la identificación y respuesta **ante distintas emergencias.**

# ALGORITMOS EMPLEADOS

## TRANSCRIPCIÓN AUDIO

Para la transcripción de audio creamos una **función en Python** que **tomaba un archivo tipo WAV** con audio en español **y lo transcribía** a texto. Para ello usamos el módulo recognize_google de la biblioteca SpeechRecognition.

## TRADUCCIÓN DE AUDIO

Decidimos **traducir el texto de español a inglés**, ya que **los modelos funcionan mejor con palabras en inglés**. Para la traducción del texto usamos la biblioteca googletrans.

## SELECCIÓN DE TOKENS

Una vez teníamos el texto teníamos que **seleccionar una serie de palabras clave (tokens)** para **reducir el tiempo de procesamiento** de dicha frase y mejorar la eficiencia de los modelos. Usamos la biblioteca nltk.

## MODELO GOOGLE VECTORES Y ANILLO DE SINÓNIMOS

Nuestro planteamiento de predicción era hacer un **comparador de palabras** usando KeyedVectors de la biblioteca gensim con el modelo GoogleNews-vectors-negative300. Para esto creamos una función que **tomaba dos palabras y devolvía un número entre -1 y 1.** Cuanto **más cercano a 1 más similitud hay** entre las palabras.

| INPUT | OUTPUT |
|---|---|
| encontrar_relacion("dog", "cat") | 0.76094574 |

**Tabla 2.** Comparación de Similitud entre Palabras utilizando KeyedVectors y GoogleNews-vectors-negative300

**En caso de que alguna de las palabras no esté en el modelo de Google** se usará la biblioteca nltk para analizar la relación entre las palabras usando el método de WordNet de **anillos de sinónimos para obtener una lista de relaciones. Cogeremos la relación máxima.**

| | |
|---|---|
| INPUT | relacion("car", "bus") |
| LIST | [0.5, 0.17, 0.17, 0.14, 0.13, 0.11, 0.1, ..., 0.06, 0.05, 0.05, 0.05, 0.05] |
| OUTPUT | 0.5 |

**Tabla 3.** Análisis de Relaciones entre Palabras utilizando WordNet y NLTK

## ANÁLISIS DE FRASE CON ESTADÍSTICO M

Para analizar una frase completa iteramos sobre los posibles casos y hacemos una predicción para cada uno.

Esta **predicción se realiza de la siguiente manera**:

1. Se **compara** cada una de las **palabras clave** asociadas al caso (10 palabras) con cada uno de los **tokens** de la frase.
2. Para cada una de las 10 palabras **se genera una lista con las relaciones**.
3. **Cogeremos (mean+max)*mean** (lo denominaremos estadístico m).
4. **Guardaremos la lista** completa para luego entrenar un modelo de **random forest que prediga mejor que el estadístico m**.

De esta manera **tendremos un valor m para cada uno de los casos.**

## MODELO RANDOM FOREST

Este modelo lo entrenamos con las listas de datos obtenidas previamente. Son **listas de longitud 10 numéricas asociadas a un booleano para poder realizar el aprendizaje supervisado**.

Este **modelo obtiene las siguientes precisiones**:

- 0.97 para la clase False
- 0.83 para la clase True

por lo que es un buen modelo para complementar al estadístico m.

## INTEGRACIÓN DE LOS MODELOS

Todos estos modelos se integran en una misma función. Esta función sigue los siguientes **pasos para analizar una determinada frase**:

1. **Traduce** la frase al inglés.
2. **Tokeniza** la frase para obtener los tokens más relevantes.
3. Para cada posible caso se **extraen las palabras clave.**
4. Para las palabras clave **extraemos la máxima relación** de cada una obteniendo una lista de 10 números.
5. **Extraemos el estadístico m** de la lista (mean+max)*mean
6. **Extraemos la probabilidad (p)** de ser True dada por el modelo de random forest.
7. **Guardamos** todo en un DataFrame de pandas.
8. **Añadiremos una nueva columna** "comb" que será la suma de m y p normalizada para que la suma de todas las clases sea igual a 1.

| key | m | p | comb |
|---|---|---|---|
| hemorrhage | 0.254323 | 0.25 | 0.232754 |
| burn intoxication inhalation | 0.115402 | 0.17 | 0.131718 |
| cardiopulmonary | 0.205002 | 0.04 | 0.113073 |
| sting | 0.098001 | 0.09 | 0.086766 |
| epilepsy | 0.122479 | 0.06 | 0.084217 |
| severe burn | 0.076565 | 0.07 | 0.067642 |
| choking | 0.117631 | 0.00 | 0.054289 |
| food poisoning | 0.116009 | 0.00 | 0.053540 |
| chemical burn | 0.114572 | 0.00 | 0.052877 |
| hypoglycemia | 0.112799 | 0.00 | 0.052059 |
| electrocution | 0.024822 | 0.06 | 0.039147 |
| fire | 0.069158 | 0.00 | 0.031918 |

**Tabla 4.** Análisis de Frases y Resultados Predictivos Integrados

9. **Controlamos las excepciones**. Todos los parámetros que se muestran a continuación han sido seleccionados tras un análisis de datos basados en frases independientes.

- **Si la suma de los estadísticos m es menor que 1.2** es porque es una frase no relacionada con ningún caso de primeros auxilios, es una frase aleatoria. En este caso se devolverá una lista vacía.
- **Seleccionaremos las variables que estén por encima de max(comb)*0.8 o de max(m)*0.8** y las devolveremos en forma de lista.

  Esto lo hacemos porque **hay situaciones de primeros auxilios que no se clasifican como una sola clase**. Por ejemplo, puede haber un incendio que conlleve quemaduras o un atragantamiento que provoque un fallo cardiopulmonar.

# CHATBOT TELEGRAM

Para crear una **interfaz de usuario fácil de utilizar y eficiente** decidimos usar Telegram. Para ello creamos un bot usando BothFather. BotFather es el método por el cual se crean todos los bot de Telegram. Esto genera un chat vacío que se puede programar accediendo con un código de API o token.

Los pasos básicos que hay que seguir para crear un bot propio con BotFather son:

1. **/start**: para iniciar la conversación con BotFather.
2. **/newbot:** para crear un nuevo bot.
3. **Crear un nombre** al bot.
4. **Crear username** del bot.

Tenemos una respuesta predeterminada asignada a cada caso.

Nuestro ChatBot **tiene las siguientes funciones** (para ver inicio de bot ir al apéndice A.1 y Apéndice A.2):

1. **Función response**: esta función genera un mensaje dada una lista de predicción.
   - **Si la lista está vacía** escribe: "Lo siento, no he podido entender lo que has dicho, ¿podrías ser más preciso?" (Véase el Apéndice A.6)
   - **Si solo hay una predicción** escribe la actuación de ese caso.
   - **Si hay dos predicciones** escribe las dos actuaciones.
   - **En caso de haber más de dos predicciones** despliega unos botones que dan las opciones al usuario para que seleccione las que quiera.

   En todos los casos **la respuesta está en forma de audio y de texto**. Además, siempre existe la posibilidad de seleccionar la opción "OTRO" dando al usuario la **opción de rectificar en caso de que se haya equivocado** al describir la situación o que el modelo haya predicho mal el tipo de caso.

2. Función message_handler: **responde a los mensajes de texto**. (Véase el apéndice A.3)
3. Función message_handler de voz: responde a los **mensajes de audio.** (Véase el apéndice A.4)
4. Función callback_query_handler: responde al **marcado de los botones desplegados.** (Véase el apéndice A.5)
5. Función crear_teclado: crea una **lista desplegable de botones**.

El ChatBot usa bot.infinity_polling() para estarse ejecutando y responder a cualquier mensaje de cualquier cliente en cualquier momento.

# ARQUITECTURA

Cuando el usuario envía un mensaje, ya sea de audio, de texto o mediante un botón, se convierte a texto. Después, se hace una predicción de la clase o clases a las que pertenece y se envía el protocolo de actuación en de esa situación.

A continuación, se muestra esto en un diagrama de flujo:

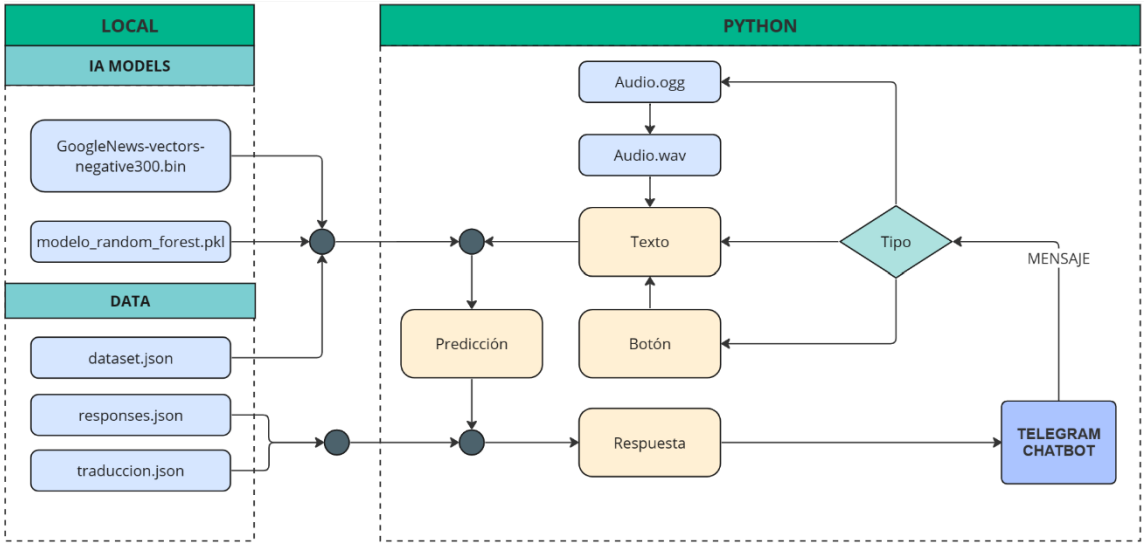



**Diagrama 1**. Procesamiento de Mensajes y Predicción de Clases para Protocolo de Actuación

# DOCKER

Para simplificar la configuración del servidor del ChatBot de Telegram, hemos desarrollado un contenedor Docker. Al ejecutarlo, el servidor estará operativo y permitirá que cualquier usuario interactúe con nuestro Bot.

Los **pasos para crear el contenedor y ejecutarlo** son:

1. docker pull raulsanzmancebo/ia_primeros_auxilios
2. docker run --rm -it raulsanzmancebo/ia_primeros_auxilios

# ACCESO USUARIO

La interfaz de usuario es muy sencilla, para empezar a usarlo habrá que descargar en PlayStore, AppStore u otras tiendas de aplicaciones la aplicación de Telegram y empezar a hablar con este chat:

https://t.me/cebollitas_healthy_ai_bot

# VENTAJAS Y DESVENTAJAS

Las **ventajas** de este proyecto son:

- El ChatBot proporciona a los usuarios **acceso inmediato** a instrucciones precisas sobre primeros auxilios, lo que puede ser vital en situaciones de emergencia donde cada segundo cuenta.
- El ChatBot está **disponible en cualquier momento del día**, lo que significa que los usuarios pueden obtener ayuda incluso fuera del horario de atención médica convencional.
- La plataforma de Telegram ofrece una **interfaz familiar y fácil de usar** para la mayoría de los usuarios, lo que facilita la comunicación y el acceso a la información necesaria.
- El modelo de inteligencia artificial detrás del ChatBot puede **adaptarse y proporcionar instrucciones específicas** para una amplia variedad de situaciones de emergencia.
- El proyecto tiene la **capacidad de escalar fácilmente** mediante la adición de nuevos tipos de casos y situaciones de emergencia.

Las **desventajas** son:

- Aunque el ChatBot puede proporcionar instrucciones básicas de primeros auxilios, puede **no ser capaz de evaluar completamente la gravedad de la situación médica**, es por esto que en casos graves siempre se remarca la importancia de avisar a emergencias o acudir a un centro médico.
- Al **depender de la tecnología y la conectividad a Internet**, el ChatBot podría no estar disponible en áreas con acceso limitado a la red, lo que podría limitar su utilidad en ciertos contextos.
- Aunque el ChatBot puede proporcionar información útil, **carece de la capacidad de empatizar o brindar apoyo emocional**, aspectos que a menudo son importantes en situaciones de emergencia.

# PLANES A FUTURO

Para mejorar aún más la eficacia y la utilidad de este proyecto, se contemplan varios planes para el futuro:

- Continuar **agregando más casos y situaciones de emergencia** a la base de datos del ChatBot, aumentando así su cobertura y relevancia.
- Refinar los algoritmos de inteligencia artificial para mejorar la precisión y la capacidad de respuesta del ChatBot, así como para **incorporar funciones adicionales, como la detección de emociones o la evaluación de la gravedad de la situación.**
- **Establecer colaboraciones con instituciones médicas**, organizaciones de salud pública y profesionales de la medicina para garantizar la precisión y la **actualización constante de la información** proporcionada por el ChatBot.
- Desarrollar funcionalidades para **personalizar la experiencia del usuario y adaptar las instrucciones proporcionadas por el ChatBo**t a las necesidades específicas de cada usuario, como la edad, el estado de salud y las preferencias individuales.

# CONCLUSIÓN

En conclusión, el desarrollo de este ChatBot de Telegram para primeros auxilios representa un paso significativo hacia la democratización del **acceso a información crucial en situaciones de emergencia médica**. A través de la integración de inteligencia artificial y la recopilación de datos confiables, hemos creado una **herramienta** que puede proporcionar **orientación rápida y precisa** para ayudar a los usuarios a **responder adecuadamente en momentos críticos**.

Es importante destacar que, si bien este ChatBot puede ser una valiosa fuente de información inicial, **no pretende reemplazar la atención médica profesional ni sustituir el juicio clínico de los profesionales de la salud**. En cambio, se concibe como un **complemento** a los servicios médicos existentes, ofreciendo un recurso adicional para el público en general y promoviendo la seguridad y el bienestar en situaciones de emergencia.

# BIBLIOGRAFÍA


Protocolos de actuación:

*Primeros auxilios*. (s. f.). https://www.uv.es/uvweb/servicio-prevencion-medio-ambiente/es/salud-prevencion/unidades/unidad-salud-laboral/primeros-auxilios-1285900419310.html

Creación de dataset de llamadas:

*ChatGPT*. (s. f.). https://chat.openai.com/

Transcripción de audio:

*SpeechRecognition*. (2023, 6 diciembre). PyPI. https://pypi.org/project/SpeechRecognition/

Traducción de audio:

*googletrans*. (2020, 14 junio). PyPI. https://pypi.org/project/googletrans/

Tratamiento del lenguaje natural:

*Gensim: topic modelling for humans*. (s. f.). https://radimrehurek.com/gensim/models/keyedvectors.html

*GoogleNews-vectors-negative300*. (2017, 11 diciembre). Kaggle. https://www.kaggle.com/datasets/leadbest/googlenewsvectorsnegative300

*NLTK :: Natural Language Toolkit*. (s. f.). https://www.nltk.org/

### Otras librerías de Python:

Ghodmode, P. (2021, 15 junio). *Telebot using Python*. DEV Community. https://dev.to/poojaghodmode/telebot-using-python-3akh

*Joblib: running Python functions as pipeline jobs — joblib 1.3.2 documentation*. (s. f.). https://joblib.readthedocs.io/en/stable/

*NumPy -*. (s. f.). https://numpy.org/

*pandas - Python Data Analysis Library*. (s. f.). https://pandas.pydata.org/

*pydub*. (2021, 10 marzo). PyPI. https://pypi.org/project/pydub/

*scikit-learn: machine learning in Python — scikit-learn 1.4.1 documentation*. (s. f.). https://scikit-learn.org/stable/

### Telegram:

*Telegram*. (s. f.). Telegram. https://web.telegram.org/a/

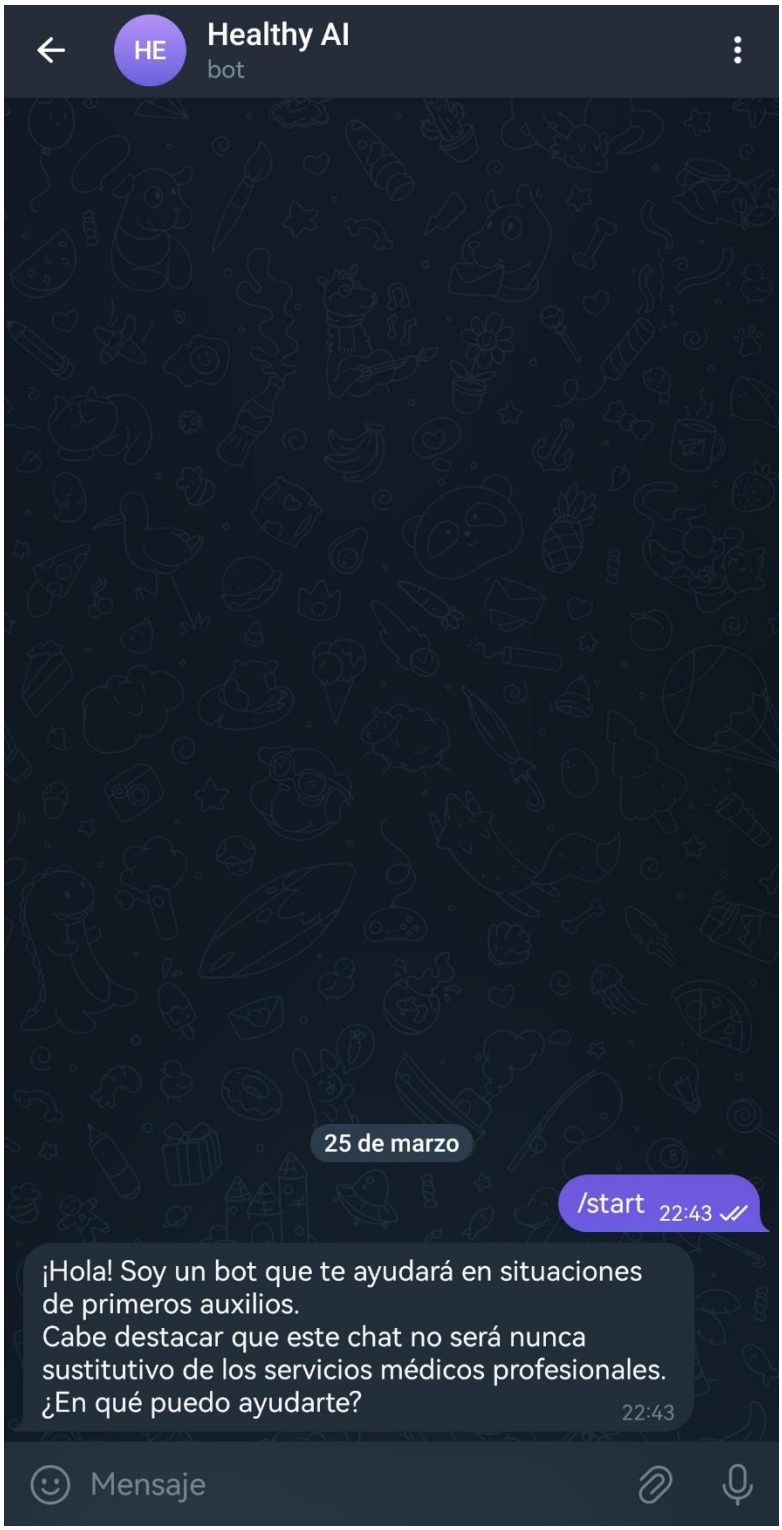
### Docker:

*Docker: Accelerated Container Application Development*. (2024, 23 enero). Docker. https://www.docker.com/

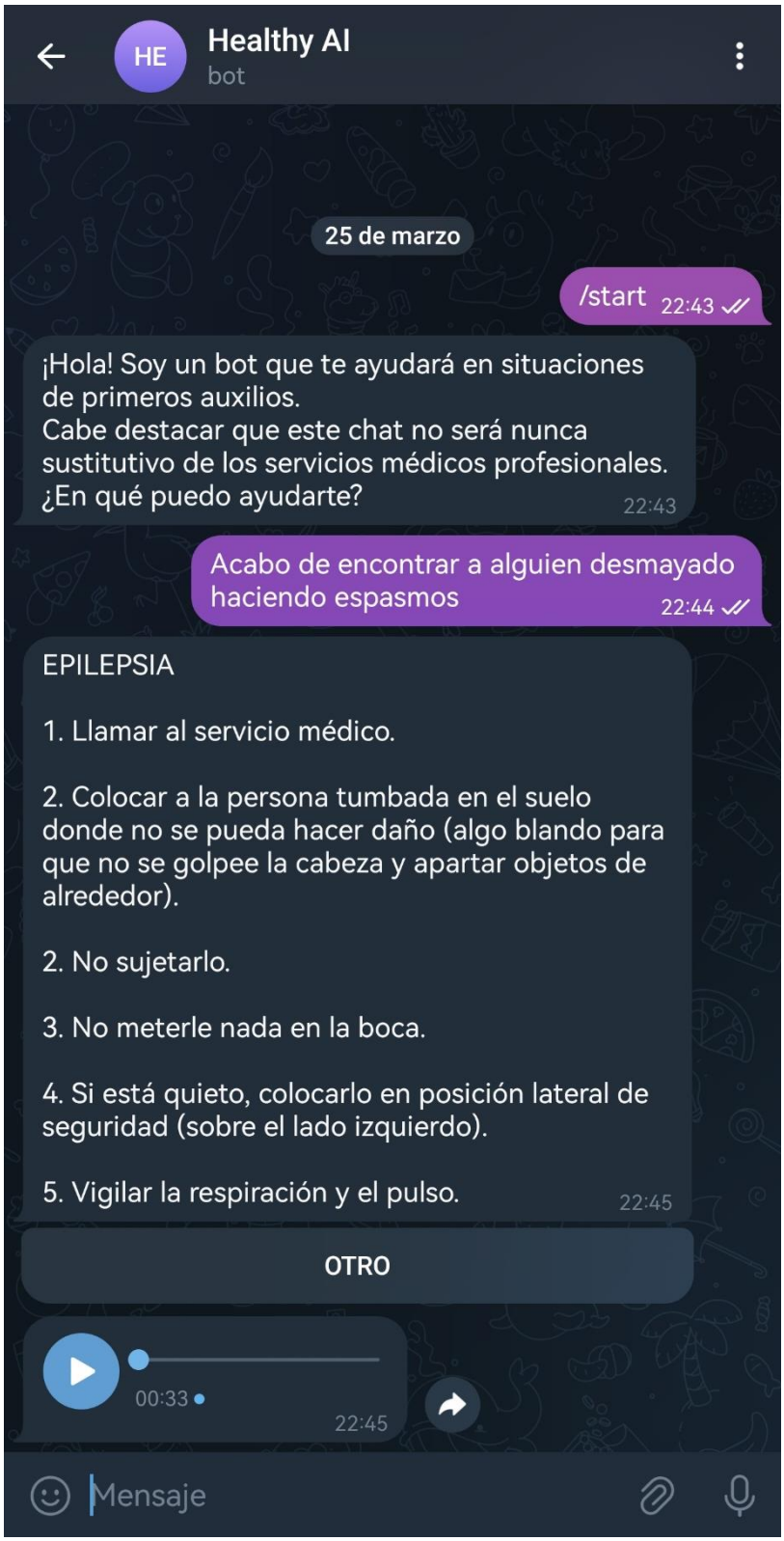
# APÉNDICE

## APÉNDICE A.1 – INICIAR CONVERSACIÓN

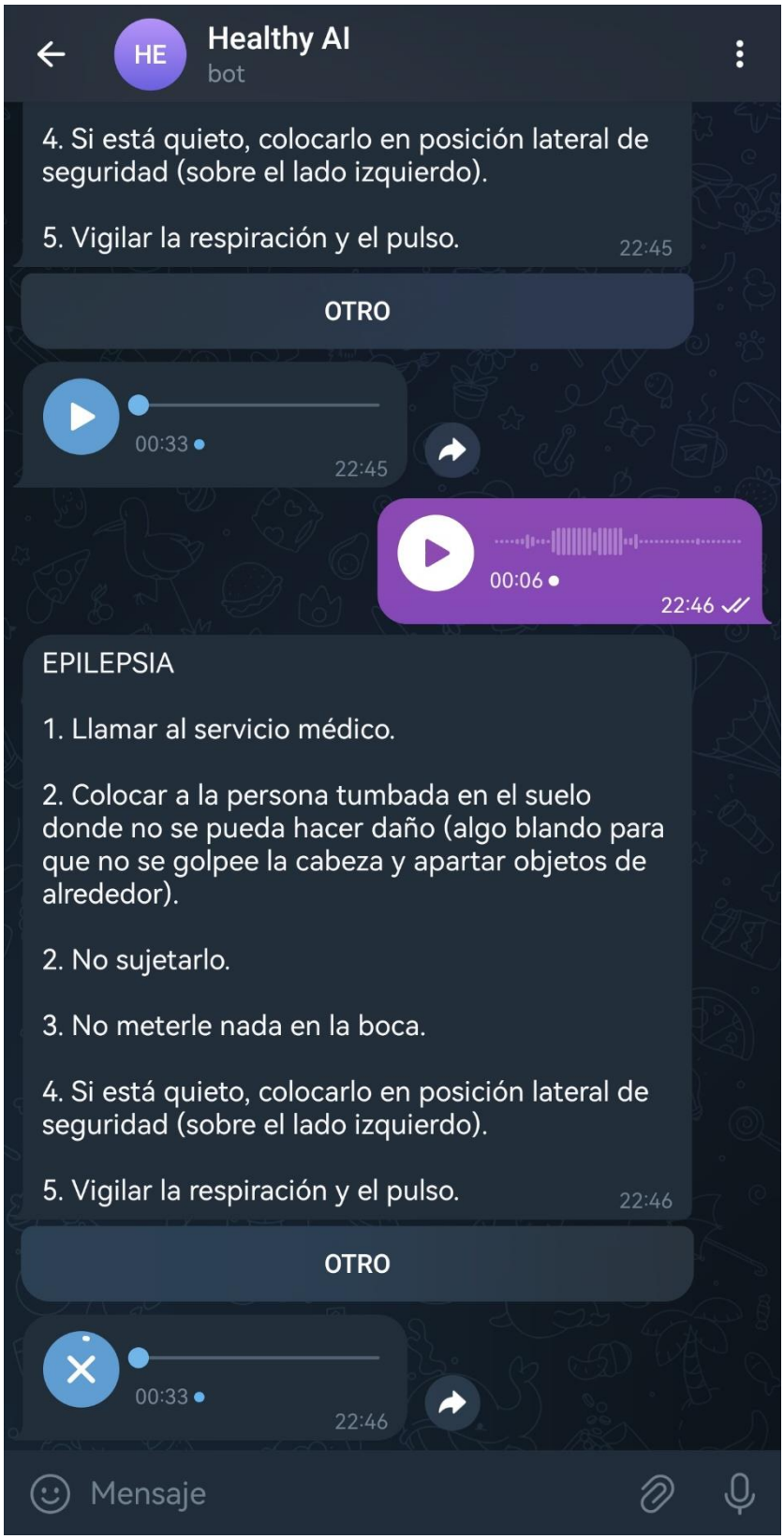
# APÉNDICE A.2 – MENSAJE DE BIENVENIDA



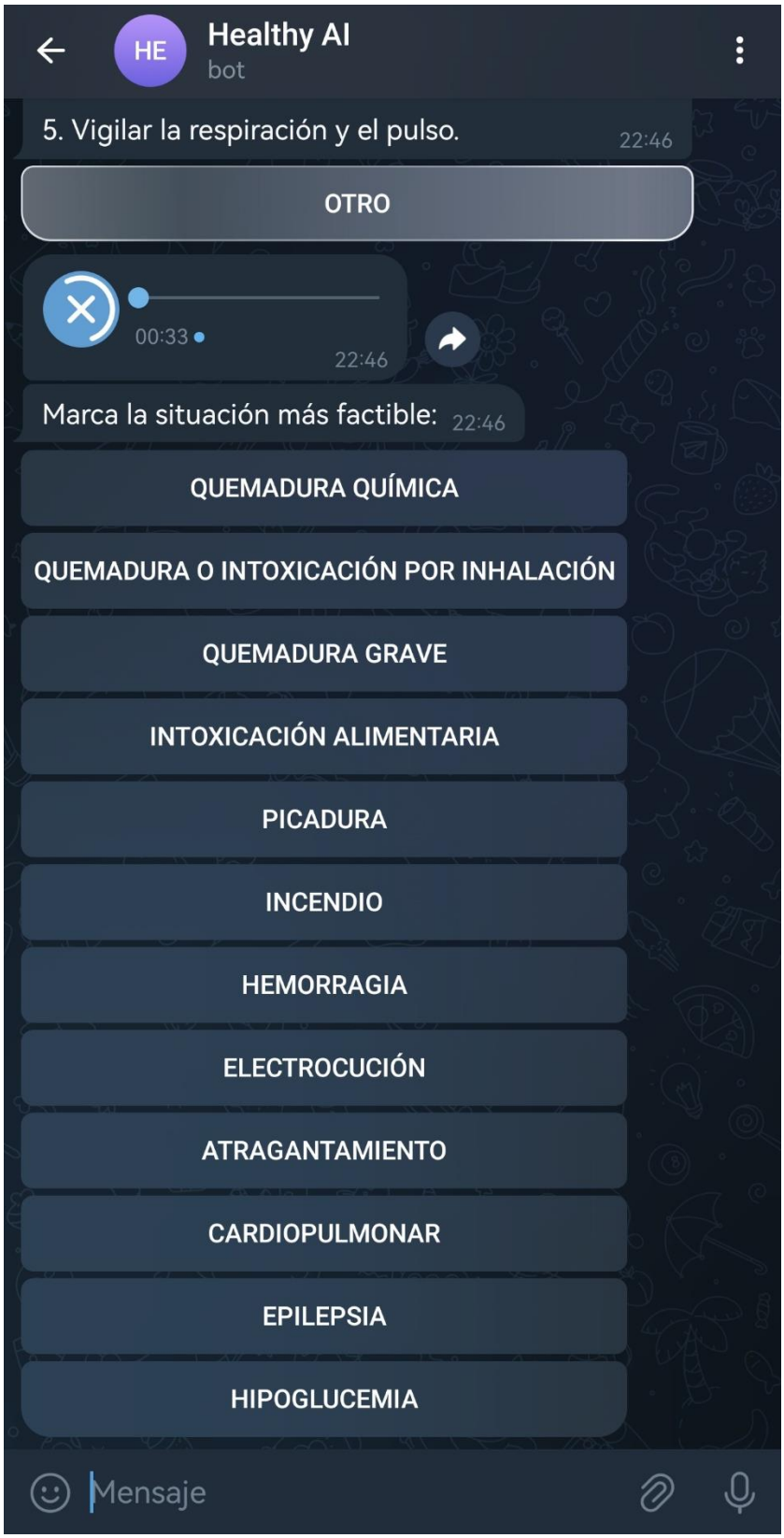
# APÉNDICE A.3 – RESPUESTA A TEXTO



# APÉNDICE A.4 – RESPUESTA A AUDIO



# APÉNDICE A.5 – SELECCIÓN DE "OTRO"



# APÉNDICE A.6 – RESPUESTA A CATEGORÍA NO ENCONTRADA