Trabajo Fin de Bootcamp en Desarrollo de Soluciones en IA junio 24

GPT-ProspectosMedicos-1.0

Pablo Salcedo

Desarrollo del proyecto

Para este proyecto se ha desarrollado un sistema que a partir de una o varias imágenes de un prospecto médico se extrae el texto de dichas imágenes y se procesa el texto para realizar una función predefinida, ya sea contestar una o varias preguntas, resumir el prospecto o extraer una serie de apartados pedidos.

Este sistema funciona principalmente con una interfaz web muy sencilla a través de la cual el usuario sube las imágenes del prospecto a la plataforma y ya el sistema por detrás se encarga de realizar todo el proceso.

Este proceso está dividido en tres partes. La parte central, el fichero **app.py** es la que controla toda la aplicación, se encarga de subir la imagen a la plataforma e ir llamando a las diferentes partes que se encargan de procesar primero la imagen y luego el texto obtenido.

En esta parte se crea el sistema usando Flask, una biblioteca para la creación de aplicaciones web [1]. Para emplear el sistema se emplea la interfaz creada en el fichero HTML del proyecto, en la carpeta templates, apoyada por la carpeta static que aloja el fichero CSS para dotar de diseño a la web. El proceso se encarga de subir la imagen a la web empleando el método POST, guardando luego estos ficheros en la carpeta indicada para ello, tras realizar diferentes verificaciones, realiza el procesamiento de la imagen con OCR devolviendo un texto que se corresponde al extraído de la imagen y guardando dicho texto en una archivo para su comprobación. Luego verifica que el texto no está vacío y si no lo está realiza el procesamiento del texto con el modelo de lenguaje. Finalmente, obtiene el resultado del modelo de lenguaje y extrae cada uno de los puntos para mostrarlos por la interfaz.

El procesamiento se ha dividido en 2 partes.

Una primera parte con un sistema OCR, que se encarga de procesar la imagen o las imágenes. Es más conveniente pasar varias imágenes para que el texto en cada una de ellas se vea lo más claro posible y la extracción sea lo mejor posible y ya el texto extraído se pasa a la segunda parte, un modelo de lenguaje que se encarga de procesar el texto y la cual es la parte más compleja debido a la longitud del texto, ya que la gran mayoría de modelos que se pueden ejecutar en local con una CPU no están diseñados para manejar tantos caracteres, pero esta parte se verá más abajo.

El sistema se dividió en 2, ya que en un primer momento se quería conseguir la interacción entre 2 modelos de IA el primero, un modelo OCR como Microsoft TrOCR, Donut o algún modelo similar, que procesara la imagen y extrajera el texto, sin embargo, ahí se encontró el primer problema, esos modelos no eran capaces de procesar imágenes con tanto texto y la extracción no era correcta aunque el prospecto se dividiera en más de 10 imágenes, por lo que

tras una investigación se encontró la herramienta Pytesseract [2] que es una herramienta de python que emplea Google-Tesseract-OCR Engine para realizar la tarea de extracción de texto de imágenes. Esta herramienta resultó ser más efectiva, ya que era capaz de extraer el texto de una cara del prospecto con una sola imagen aunque con algún fallo, por lo que sigue siendo recomendable subir varias imágenes enfocándose en las partes de las que se quiere extraer información y luego el modelo ya se encarga de concatenar las diferentes partes leídas.

Para poder realizar este proyecto, primero se necesitaban datos. Tanto para comprobar que el modelo OCR extraía el texto correctamente como datos que pasarle al modelo de lenguaje para ir analizando su funcionamiento. Para ello se realizaron fotografías a diferentes prospectos médicos que tenía en casa, una vez se consiguieron las imágenes de varios prospectos era el momento de evaluar la calidad de la extracción, y dado que no era muy complejo se realizó manualmente, comparando con el prospecto que se puede encontrar en la web de CIMA de la AEMPS(Agencia Española de Medicamentos y Productos Sanitarios) [3], esto también permitió ver que a pesar de que la extracción era bastante buena, algunos caracteres o palabras no las reconocía bien en algunos momentos y en algunos saltos de línea incluía algunos caracteres extraños en algunas imágenes, por lo que también fue una buena oportunidad para identificar qué tipo de limpieza se iba a tener que aplicar al texto antes de procesarlo con el modelo de lenguaje.

Para esta primera parte del modelo, el sistema OCR, tenemos el fichero modeloOCR_tesseract.py. Para ello tenemos una única función que recibe por parámetro las imágenes que el usuario sube a través de la web y una carpeta donde el sistema guardará las imágenes procesadas para poder verificar lo que gestiona el sistema. Se probó a realizar transformaciones en la imagen para intentar mejorar la calidad de la lectura, pero ninguna tuvo el efecto deseado por lo que a la herramienta pytesseract se le pasa la imagen como se sube a la plataforma. Luego realiza la lectura de la imagen, en el caso de que sean varias, leerá una a una y al final concatenará el resultado para formar el texto completo. Se han puesto varios puntos de control en el código por si fallase el procesamiento, la imagen no fuese correcta por el formato o la extensión no fuese correcta. También se guarda el texto extraído en un fichero de texto para verificar lo que se ha extraído y comprobar su calidad.

La segunda parte es la que se encarga de procesar el texto extraído anteriormente y realizar lo que se le pida por un prompt. En esta primera versión viene predefinido en el código, en este caso, es que extraiga el nombre del medicamento, el principio activo, la dosis recomendada o

manera de tomarlo y los posibles efectos secundarios o adversos y el sistema al final saque esta información por la interfaz web. El prompt empleado ha sido el siguiente:

Aquí tienes un prospecto médico extraído por OCR:

{prospecto}

Por favor, extrae la siguiente información y devuélvela en este formato:

- Nombre del medicamento: [Nombre del medicamento].

- Principio activo: [Principio activo del medicamento].

- Dosis recomendada: [Dosis recomendada para el medicamento].

- Posibles efectos adversos: [Lista de posibles efectos adversos].

Si no encuentras información en alguna categoría, déjalo vacío.

"""

Sin embargo, como ya se ha mencionado, esta parte es la más compleja, ya que muchos de los modelos no están capacitados para procesar textos tan largos, muchos de ellos tienen un límite de 512 tokens o caracteres. Por esto ha sido necesario dividir el texto en las diferentes secciones que componen un prospecto médico. Esta división se ha realizado empleando la librería RegEx, para identificar las diferentes secciones identificadas por un número, por ejemplo "1. Qué es Ebastel Forte y para qué se utiliza".

Aunque primero se ha realizado una limpieza del texto y tras realizar diferentes pruebas, lo mejor fue eliminar solamente los espacios en blanco extras, ya que eliminar más cosas terminó siendo más problemático que beneficioso, ya que se perdía información.

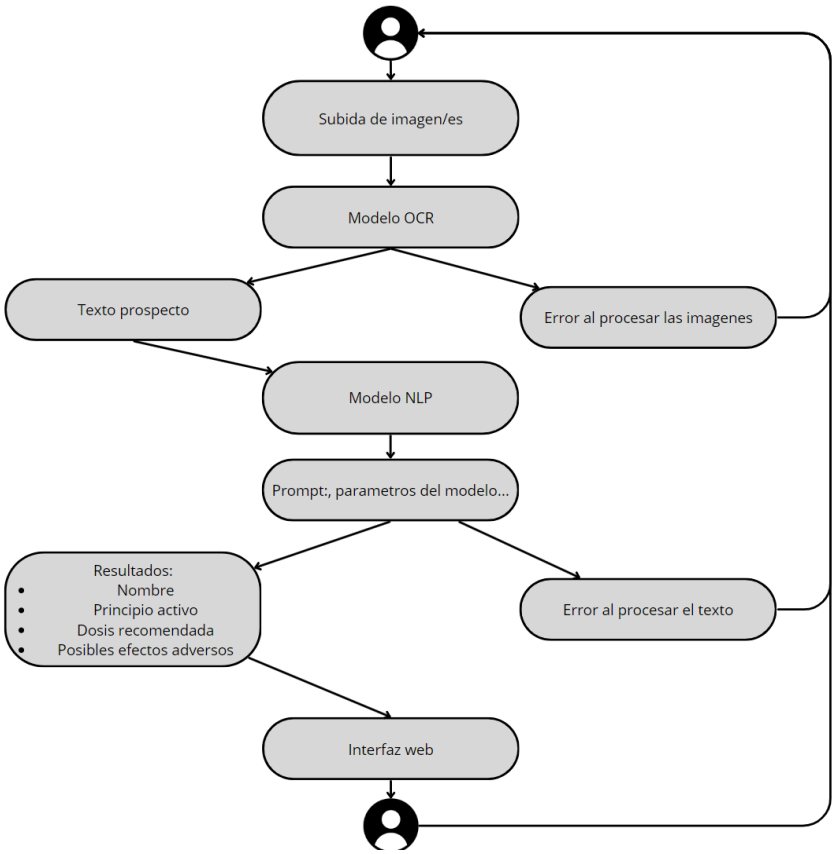
En un primer momento, se realizó la división en fragmentos de 512 tokens, que es el máximo que puede procesar el modelo empleado, pero había un problema muy importante y es que se dividían secciones por lo que se perdía contexto y mucha información, quedando la extracción de por ejemplo los efectos adversos a medias.

Para la parte de procesamiento del texto, en el fichero modeloNLP.py se ha empleado el modelo **Google/flan-t5-large**[4], se ha elegido este modelo después de probar otros diferentes que ofrecieron resultados tanto similares como peores en algunos casos, pero este era un modelo con buenos resultados y un tiempo de ejecución aceptable. Esta parte del sistema consta de 6 funciones diferentes.

- La primera, **limpiar_texto**, elimina los espacios en blanco dobles y los espacios del principio y el final de la cadena de texto con la función strip.
- La siguiente función, **dividir_secciones**, se encarga de dividir el texto en las diferentes secciones que componen un prospecto médico. Emplea la función RegEx buscando un número seguido del inicio del nombre de las secciones. Luego crea un

diccionario donde se almacenan las distintas secciones donde cada título va asociado al contenido de la sección.

- La siguiente función, **procesar_sección** es en la que se procesa cada sección usando el modelo y el prompt. Primero revisa que la sección no esté vacía y si no lo está procesa la sección correspondiente.
- A la hora de devolver la respuesta, se emplea la función **extraer_información** sobre la respuesta generada. En esta función extrae el valor de cada uno de los apartados pedidos. Se emplea de nuevo la librería RegEx para busca los patrones que identifiquen cada parte. La más compleja es la de los efectos adversos, ya que se dividió según la categoría de la frecuencia de los efectos empleada en los prospectos, por ejemplo 'Muy frecuente', 'Raros', 'Frecuencia desconocida'. Devuelve un diccionario con toda la información asociada a su etiqueta.
- Con la función de **consolidar_resultados** se combinan los resultados obtenidos en cada sección para cada uno de los apartados pedidos y se crea un diccionario para almacenar toda esta información.
- Finalmente, la función **procesar_texto_completo** se encarga de ir llamando al resto, es la que devuelve los resultados finales que serán los que se publicarán en la interfaz.



Líneas futuras de trabajo

De cara al futuro este sistema se podría seguir ampliando y mejorando, por ejemplo, la ampliación más sencilla que no se ha realizado en este momento es mejorar la interfaz, ya que no era el objetivo en este trabajo no se ha realizado una interfaz mejor, pero se podría realizar una interfaz mucho más user-friendly, con más color, una sección de ayuda para que el usuario sepa como usar la web y que esperar de ella.

La mejora más importante sería mejorar el sistema, tanto la parte OCR para mejorar la extracción de texto, ya que debido a la variedad de prospectos, su tipo de letra y formato, el texto leído es muy variado y sobre todo mejorar el procesamiento del texto para preparar el modelo para manejar todo tipo de prospectos.

La siguiente mejora que no sería muy compleja de implementar sería la de permitir que sea el usuario quien escriba el prompt para poder indicar y especificar que información desea obtener acerca del prospecto que está tratando, lo cual sería una característica que aumentaría la satisfacción del usuario al ser un sistema más único y personal para cada usuario. Para ello se debería mejorar el procesamiento del texto extraído para que maneje lo que se pida por prompt.

Un aspecto que mejoraría la experiencia de uso considerablemente es crear una aplicación móvil que sirva para usar el sistema. La aplicación permitiría realizar fotos o seleccionar fotos de la galería para ser procesadas por el modelo en lugar de funcionar solamente a través de una web.

También se podrían llevar a cabo nuevas funcionalidades que permitan diferentes usos, por ejemplo, si el sistema detecta correctamente el nombre del medicamento, podría almacenar la información de ese medicamento en una base de datos y que en el futuro el usuario pueda seleccionar directamente el medicamento por su nombre desde una lista sin tener que subir una fotografía, es decir, si un usuario sube una foto del prospecto de Aerius y el sistema reconoce correctamente el nombre podría almacenar toda la información en una base de datos e ir actualizándola con nueva información y en el futuro permitir que los usuarios seleccionen medicamentos ya guardados y no tengan que realizar y subir una fotografía, es decir construir un sistema RAG, que permita diferentes tipos de interacción de los usuarios. Otra mejora que también sería importante sería la unificación de las tareas del sistema, es decir que la parte OCR y el modelo de lenguaje en una arquitectura unificada. Esto sería capaz con un modelo como Donut [5], LayoutLMv3 [6] o GPT-4 de OpenAI [7].

Finalmente, la última línea de trabajo futura sería convertir el sistema en un sistema abierto a todo tipo de documentos, es decir, poder pasar fotos de todo tipo de documentos y que a

través del prompt el usuario indique que información desea. Por ejemplo para recetas de cocina, el usuario sube una foto y quiere que le indique los ingredientes exactos y los pasos a seguir de manera esquemática, o por ejemplo el usuario sube una foto de unas instrucciones y desea que se las resuma paso a paso en lugar de tener que leerse toda la página, esto podría convertir el sistema en una herramienta global de extracción de puntos clave o resumen de documentos, lo cual permitiría a las personas ahorrar tiempo al no tener que leer el documento completo.