Deusto
Universidad de Deusto
Deustuko Unibertsitatea

MUCSI - Deep Learning + Minería de texto y procesamiento de lenguaje natural

Herramienta de clasificación y generación de textos de pacientes con enfermedades mentales mediante Deep Learning y NLP

Trabajo final

Unai Sainz, Lander Hernández e Iker Silva

26-05-2023

ÍNDICE

1.- INTRODUCCIÓN

Hoy en día, tan solo en España, existen más de 500.000 personas que padecen de desórdenes mentales y neurológicos[1]. Según Anisalud, los trastornos mentales afectan a un 19,5% de la población española, y en todo el mundo, se prevé que 1 de cada 4 personas tendrá un trastorno mental a lo largo de su vida[2]. Estas enfermedades abarcan el 40% de las enfermedades crónicas españolas, y son la mayor causa de los pacientes que viven durante años con discapacidad.

Actualmente, el proceso que se sigue en el ámbito sanitario para diagnosticar a un paciente de una enfermedad mental parte de una evaluación inicial con el profesional. El objetivo de esta primera entrevista es conocer su historial, síntomas, y factores relevantes para el diagnóstico. Después, se hace una evaluación psicológica exhaustiva, donde se realizan pruebas como cuestionarios, test de personalidad y evaluaciones neuropsicológicas. Por último, se analizan y evalúan los datos recopilados hasta llegar a un diagnóstico.

Sin embargo, el proceso de evaluación puede resultar difícil en muchos casos, dando lugar a un diagnóstico erróneo. En muchos casos, los síntomas de las enfermedades mentales pueden solaparse con otras muchas enfermedades, lo que dificulta un diagnóstico preciso. Además, muchos pacientes pueden ocultar los síntomas, o tienen dificultades para mostrarlos.

Otro de los factores que dificulta el diagnóstico, es la falta de pruebas objetivas[3]. Como se ha mencionado anteriormente, las pruebas consisten en entrevistas y cuestionarios simples. El tiempo dedicado a la entrevista, y la experiencia del entrevistador son hasta ahora factores clave para poder llegar a un correcto diagnóstico.

Mediante este proyecto se pretende crear una herramienta de apoyo en el diagnóstico de enfermedades mentales. Esta solución puede aportar a los diagnósticos un grado de objetividad que haga que un paciente sea diagnosticado en un tiempo menor y con un mayor grado de confianza.

El objetivo de este proyecto es desarrollar una herramienta de apoyo en el diagnóstico de enfermedades mentales que mejore la objetividad, acelere los tiempos de diagnóstico y aumente la confianza en los resultados.

---

[1] Fuente: https://www.anisalud.com/actualidad/notas-de-prensa-anis/3216-mas-de-medio-millon-de-personas-en-espana-padecen-desordenes-mentales-y-neurologicos

[2] Fuente: https://www.anisalud.com/actualidad/notas-de-prensa-anis/3216-mas-de-medio-millon-de-personas-en-espana-padecen-desordenes-mentales-y-neurologicos

[3] Fuente: https://www.menteamente.com/blog-salud-mental/psiquiatra-segunda-opinion

2.- ESTADO DEL ARTE

2.1.- MÉTODOS DE TRATAMIENTO DE DATASET

En cuanto al estado del arte de los métodos de tratamiento de datasets con características similares al seleccionado, es decir, los conjuntos de datos relacionados con la salud mental extraídos de plataformas de redes sociales, como Reddit, ha sido objeto de una creciente investigación en el campo del Procesamiento de Lenguaje Natural. El NLP se ha convertido en una herramienta fundamental para comprender y abordar la gran cantidad de información textual generada por los usuarios en estas plataformas.

Uno de los principales desafíos en el tratamiento del conjunto de datos "Reddit Mental Health Dataset" es la comprensión y clasificación precisa del contenido textual en términos de emociones, sentimientos y temas relacionados con la salud mental. Los investigadores han aplicado técnicas de análisis de sentimientos y emociones para identificar la polaridad emocional en los mensajes, permitiendo una mejor comprensión de los estados emocionales de los usuarios. Esto ha permitido descubrir patrones y tendencias emocionales en diferentes comunidades y subreddits, lo que ayuda a identificar la prevalencia de emociones positivas o negativas en relación con la salud mental.

Además, se han desarrollado modelos de clasificación de texto y técnicas de aprendizaje automático para identificar y categorizar mensajes relacionados con temas específicos de salud mental, como depresión, ansiedad, trastornos alimentarios y trastornos del estado de ánimo. Estos modelos han demostrado ser útiles para detectar señales de riesgo y proporcionar intervenciones tempranas y apoyo a los usuarios que podrían estar experimentando problemas de salud mental.

Otra área de investigación destacada en el estado del arte del tratamiento del conjunto de datos "Reddit Mental Health Dataset" es la detección de riesgo y prevención de crisis. Los investigadores han utilizado técnicas de NLP para desarrollar algoritmos que pueden identificar indicios de suicidio, autolesiones u otros comportamientos de riesgo en los mensajes de los usuarios. Esto ha permitido el desarrollo de sistemas de detección temprana de crisis y la posibilidad de proporcionar recursos de apoyo y asistencia a los usuarios en momentos críticos.

Además, el análisis de la estructura de la red social en Reddit ha sido un enfoque importante en el estado del arte. El estudio de las interacciones entre los usuarios y la difusión de información relacionada con la salud mental ha permitido identificar comunidades de apoyo, influenciadores y la propagación de información errónea. El análisis de redes sociales ha proporcionado una comprensión más profunda de cómo se forman y desarrollan las comunidades en línea y ha permitido la identificación de estrategias efectivas para difundir información precisa y fomentar la participación en discusiones saludables sobre la salud mental.

En términos de privacidad y ética, los investigadores también han enfocado sus esfuerzos en proteger la privacidad de los usuarios y garantizar el consentimiento informado. Se han implementado técnicas de anonimización de datos y se ha trabajado en el desarrollo de

marcos éticos para el manejo responsable de los conjuntos de datos relacionados con la salud mental.

2.2.- MODELOS DE LENGUAJE

Un modelo de lenguaje es un tipo de modelo estadístico o de aprendizaje automático que se utiliza para predecir la probabilidad de ocurrencia de una secuencia de palabras en un idioma determinado. Estos modelos son fundamentales en el campo del procesamiento del lenguaje natural y se aplican en una amplia variedad de tareas, como la traducción automática, la generación de texto, la corrección ortográfica y la respuesta a preguntas, entre otras.

Existen diferentes tipos de modelos de lenguaje, pero uno de los enfoques más populares y exitosos es el basado en redes neuronales, específicamente en redes neuronales recurrentes (RNN) y transformers.

Las RNN son un tipo de red neuronal que se utiliza en modelos de lenguaje secuenciales, ya que son capaces de capturar la dependencia temporal entre las palabras en una secuencia. Estas redes procesan una secuencia de palabras una por una, utilizando información contextual de palabras anteriores para predecir la siguiente palabra en la secuencia.

Los transformers, por otro lado, son una arquitectura de red neuronal más reciente y poderosa utilizada en modelos de lenguaje. Estos modelos basados en atención son capaces de capturar relaciones a largo plazo en el texto y procesar las palabras en paralelo, en lugar de secuencialmente. Esto los hace especialmente adecuados para el procesamiento de texto a gran escala.

Los modelos de lenguaje se entrenan utilizando grandes conjuntos de datos de texto en el idioma de interés. Durante el entrenamiento, el modelo aprende a asignar probabilidades a las palabras y secuencias de palabras en función del contexto y la distribución estadística en el conjunto de datos de entrenamiento. El objetivo es maximizar la probabilidad de la secuencia correcta y minimizar la probabilidad de secuencias incorrectas.

Una vez entrenados, los modelos de lenguaje se pueden utilizar de varias formas. Pueden generar texto automáticamente, completar oraciones o predecir la siguiente palabra dada una secuencia de palabras. También se pueden utilizar en tareas de traducción automática, donde se utilizan modelos de lenguaje para estimar la probabilidad de una traducción dada una secuencia de palabras en el idioma de origen.

Además, los modelos de lenguaje pre-entrenados, como BERT y GPT, han demostrado ser extremadamente útiles. Estos modelos se entrenan en grandes cantidades de datos sin etiquetar y luego se ajustan a tareas específicas con conjuntos de datos más pequeños. Esto permite que los modelos utilicen su conocimiento pre-entrenado para mejorar el rendimiento en tareas específicas con un menor requerimiento de datos de entrenamiento.

2.2.1.- BERT

BERT (Bidirectional Encoder Representations from Transformers) es un modelo de lenguaje basado en redes neuronales de transformers que ha revolucionado el campo del procesamiento del lenguaje natural. Fue presentado por Google en 2018 y ha demostrado un rendimiento sobresaliente en una amplia gama de tareas de NLP, incluyendo el reconocimiento de entidades, la clasificación de texto, el etiquetado de partes del discurso y la respuesta a preguntas, entre otros.

La característica principal de BERT es su capacidad para comprender el contexto en el que se encuentra una palabra en una oración, utilizando información de las palabras que la rodean tanto a la izquierda como a la derecha. Esto se logra mediante un proceso de pre-entrenamiento y afinamiento, que implica entrenar el modelo en grandes cantidades de texto sin etiquetar y luego ajustarlo a tareas específicas mediante un entrenamiento supervisado.

Durante el pre-entrenamiento, BERT utiliza una tarea de "llenado de huecos" conocida como Masked Language Modeling (MLM). En esta tarea, se enmascaran aleatoriamente algunas palabras de una oración y el modelo debe predecir las palabras enmascaradas en función del contexto circundante. Además, se introduce una tarea de "siguiente oración" para que el modelo aprenda a entender la relación entre dos oraciones consecutivas.

Después del pre-entrenamiento, BERT se ajusta a tareas específicas mediante un proceso de afinamiento o fine-tuning. Se alimenta al modelo con ejemplos etiquetados de la tarea en cuestión y se ajustan los pesos de las capas superiores para adaptarse a la tarea específica. Este afinamiento permite que BERT utilice su conocimiento pre-entrenado y mejore su desempeño en tareas específicas con conjuntos de datos más pequeños.

Una de las principales ventajas de BERT es su capacidad para capturar el significado contextual de las palabras. Al considerar tanto el contexto a la izquierda como a la derecha de una palabra, BERT es capaz de comprender mejor las ambigüedades y polisemias en el lenguaje humano. Esto ha llevado a mejoras significativas en muchas tareas de procesamiento del lenguaje natural, superando a los enfoques tradicionales y estableciendo nuevos puntos de referencia en el campo.

Sin embargo, BERT también tiene algunas limitaciones. Debido a su tamaño y complejidad, es computacionalmente costoso y requiere una gran cantidad de recursos de hardware para entrenar y utilizar. Además, BERT trata el texto como una secuencia de palabras y no captura relaciones más complejas a nivel de frases o párrafos.

A pesar de sus limitaciones, BERT ha sentado las bases para una nueva generación de modelos de lenguaje pre-entrenados y ha impulsado importantes avances en el campo del procesamiento del lenguaje natural. Su enfoque basado en transformers ha demostrado ser altamente efectivo en el procesamiento de texto y ha llevado a mejoras significativas en una amplia gama de aplicaciones de NLP.

2.2.2.- BigBird

BigBird es un modelo de lenguaje basado en transformers que se desarrolló como una extensión de BERT. Fue presentado por Google en 2020 y se diseñó específicamente para abordar el desafío de procesar secuencias más largas de texto de manera más eficiente.

A diferencia de los modelos de lenguaje tradicionales, que procesan las secuencias de texto de manera secuencial, BigBird utiliza una estrategia de atención global y localizada para capturar las relaciones entre las palabras en el contexto de una secuencia más larga. Esto significa que BigBird puede considerar la información contextual no solo de las palabras cercanas, sino también de las palabras distantes dentro de una secuencia.

El enfoque clave de BigBird es la utilización de una matriz de ataque estructurada (Structured Attention Matrix), que permite que el modelo solo atienda a una fracción de las palabras en una secuencia, en lugar de considerar todas las interacciones posibles. Esto reduce drásticamente los requisitos computacionales y de memoria, lo que permite el procesamiento eficiente de secuencias largas.

La estructura de atención de BigBird se basa en dos tipos de atención: la atención global y la atención local. La atención global permite que el modelo capture las dependencias a largo plazo en una secuencia, mientras que la atención local se enfoca en las interacciones cercanas. Esto proporciona un equilibrio entre la capacidad de capturar el contexto de largo alcance y la eficiencia computacional.

Otra característica importante de BigBird es la utilización de patrones de distribución de tipo "*sparse*" en la matriz de ataque estructurada. Esto significa que el modelo solo atiende a una selección de palabras de la secuencia en cada paso, en lugar de todas las palabras. Esto reduce aún más la complejidad computacional y de memoria.

El enfoque innovador de BigBird ha demostrado ser altamente efectivo en el procesamiento de secuencias largas de texto, superando las limitaciones computacionales de los modelos anteriores. Ha logrado resultados sobresalientes en diversas tareas de procesamiento del lenguaje natural, como la traducción automática, la generación de texto y la comprensión de preguntas y respuestas.

Si bien BigBird ha demostrado ser un avance significativo, también tiene algunas limitaciones. Debido a su estructura más compleja y su requisito de memoria adicional para almacenar la matriz de ataque estructurada, BigBird sigue siendo computacionalmente costoso en comparación con modelos más simples como BERT. Esto puede limitar su adopción en entornos con recursos limitados.

2.2.3.- GPT-2

GPT-2 (Generative Pre-trained Transformer 2) es un modelo de lenguaje basado en transformers desarrollado por OpenAI y presentado en 2019.

GPT-2 se basa en la arquitectura de transformers, que es un tipo de red neuronal que utiliza la atención para capturar las relaciones entre las palabras en un texto. A diferencia de los modelos anteriores que procesan el lenguaje de manera secuencial, GPT-2 utiliza la atención para capturar conexiones a largo plazo en el texto, lo que le permite comprender mejor el contexto y generar texto coherente y relevante.

Lo que hace que GPT-2 sea particularmente impresionante es su capacidad para generar texto de alta calidad de manera autónoma. El modelo se entrena en un gran corpus de texto sin supervisión, utilizando un proceso conocido como pre-entrenamiento. Durante el pre-entrenamiento, el modelo aprende a predecir la siguiente palabra en una secuencia de texto dada su historia previa.

Después del pre-entrenamiento, GPT-2 se puede ajustar a tareas específicas mediante un proceso de afinamiento o fine-tuning. Se le proporciona un conjunto de datos específico para una tarea determinada y se ajustan los pesos del modelo para adaptarse a esa tarea en particular. Esto permite que GPT-2 se utilice para una amplia gama de aplicaciones, como la generación de texto, la traducción automática, el resumen de texto y la respuesta a preguntas, entre otros.

GPT-2 ha demostrado una notable habilidad para generar texto coherente y convincente en una variedad de dominios. Sin embargo, también ha planteado preocupaciones éticas y de seguridad debido a su potencial para generar contenido falso o engañoso. Debido a estas preocupaciones, OpenAI decidió inicialmente no publicar la versión completa de GPT-2 para evitar un mal uso potencial. Sin embargo, versiones más pequeñas del modelo se han puesto a disposición del público.

3.- DOCUMENTACIÓN DE LA ARQUITECTURA PROPUESTA

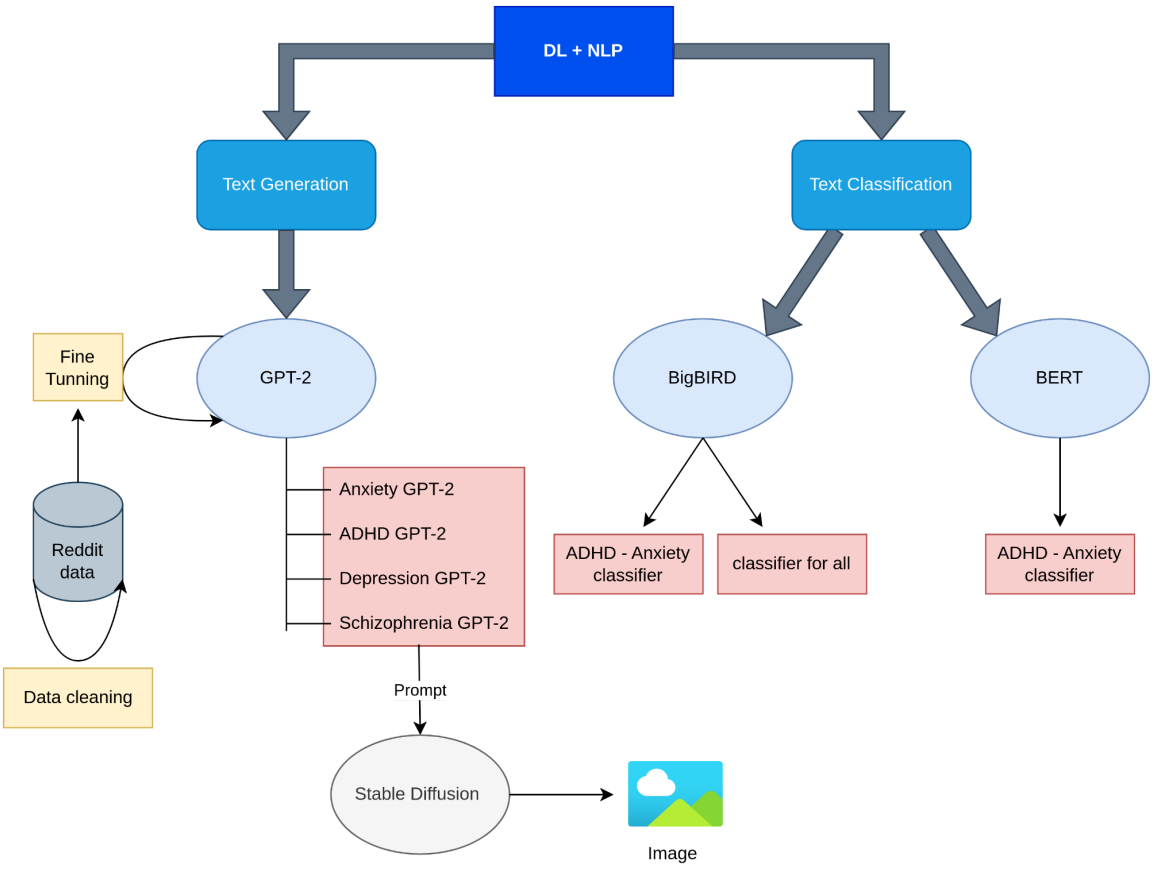
El presente proyecto, el cual abarca las asignaturas de Deep Learning y minería de texto y procesamiento de lenguaje natural, está dividido en 2 apartados principales.

Por un lado, tenemos la parte de clasificación de textos, en la que se han entrenado 2 modelos con únicamente 2 posibles outputs o condiciones mentales para comprobar su funcionamiento. Una vez se ha comprobado que el BigBIRD funcionaba mejor que BERT, se ha entrenado de nuevo el primer modelo referenciado, esta vez con todas las posibles condiciones mentales contenidas en el dataset utilizado.

Por otra parte, se ha desarrollado la parte de generación de texto. En este caso, se han entrenado varios modelos GPT-2, cada uno con corpus generados a partir de una condición mental específica. En específico, se han obtenido GPT-2 especializados para ansiedad, ADHD, depresión y esquizofrenia.

Además de todo lo mencionado, se ha optado por implementar una funcionalidad de generación de imágenes mediante stable diffusion, mediante la cual se han generado múltiples imágenes mediante diversos prompts relacionados con cada una de las condiciones tratadas.

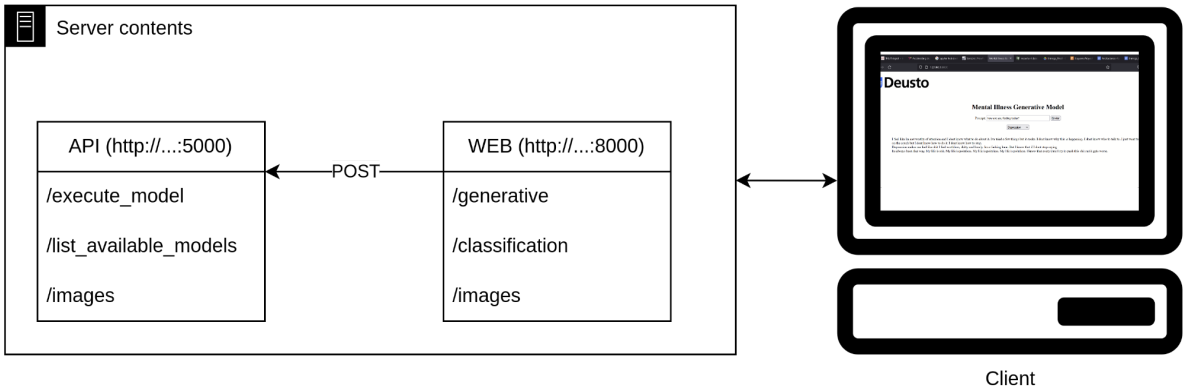
A continuación se muestra un esquema general del presente trabajo:



Se ha decidido dar vida al proyecto mediante el despliegue del mismo en un servidor, con intención de que cualquier usuario inexperto en el ámbito de programación pueda utilizar el servicio.

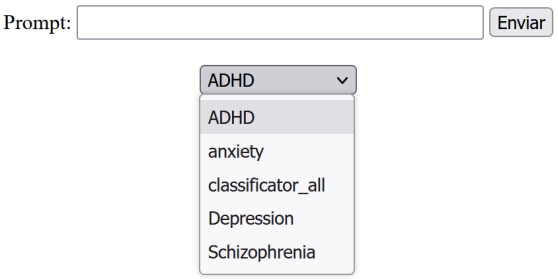
Para el despliegue del proyecto se ha visto conveniente utilizar Docker de forma que no afecte a la compatibilidad entre el resto de programas en ejecución ya existentes en el servidor. Docker también facilita el despliegue de la aplicación, al tratarse de programas que se ejecutan bajo contenedores.

Los componentes de la aplicación creada vienen definidos en la siguiente ilustración.



Como se puede apreciar, consiste en dos módulos: el frontend (WEB)y el backend (API). El frontend contiene 3 páginas HTML desde las que se puede interactuar con el trabajo desarrollado en el proyecto. El backend consiste en una API que se encarga de direccionar las peticiones al código correspondiente. Son 3 los servicios que presta la interfaz:

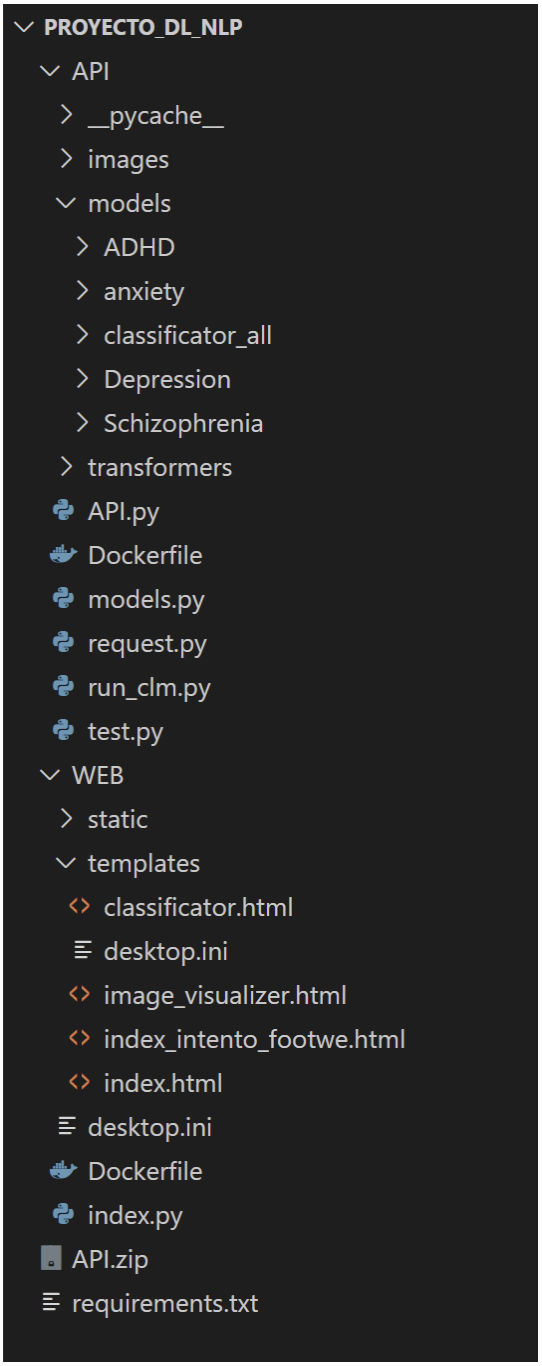
- ***execute_model*** (POST): ejecuta el modelo generativo con el prompt de entrada, al modelo seleccionado por el usuario.
- ***list_avilable_models*** (POST): función necesaria para que el frontend pueda mostrar el listado de modelos disponibles. El código se ocupa de contar los modelos que existen en la carpeta del proyecto y devolver la información de la disponibilidad al usuario.



El resultado se muestra, como se puede ver en la imagen anterior, en el desplegable de modelos.

- ***images*** (GET): api que devuelve una imagen de la enfermedad seleccionada.[4]
- ***execute_classification_model*** (POST): ejecuta el modelo de clasificación, recibiendo un texto de entrada.

El código fuente se estructura de la siguiente manera.



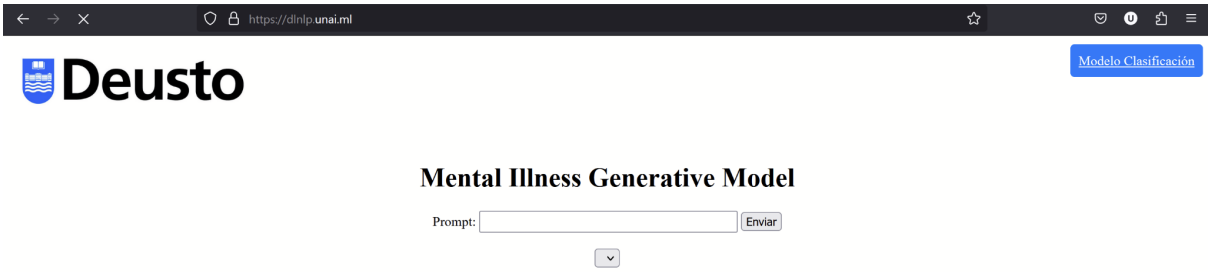
Como se puede observar, el código se divide en las mismas secciones mencionadas en los componentes de la aplicación, que se ejecutan individualmente en paralelo.

- WEB:

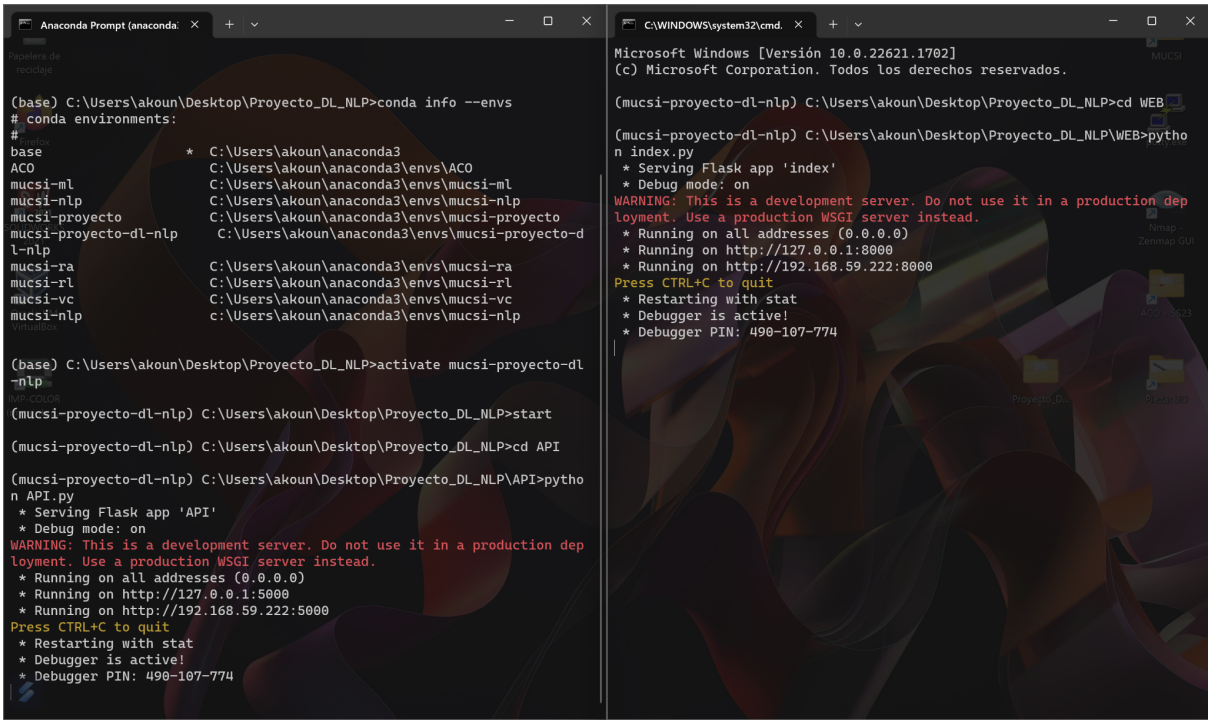
[4] Función no integrada con el frontend en la versión presentada en la asignatura.

    - Se trata del FrontEnd del programa. Alberga las imágenes, hace las peticiones a la API y muestra los resultados por pantalla. Es la parte visual con la que interactúa el usuario.
- API:
    - Es el BackEnd que se encarga de recibir las peticiones, procesar la solicitud, generar una salida y enviarla. Es donde se ejecutan los modelos de IA.

Por motivos explicados en el último capítulo del documento, se ha decidido diseñar una interfaz sencilla y funcional. Consiste en un campo de entrada de texto junto a un botón de enviar, el cual hace la petición a la API. También hay un botón que redirige a la página de la segunda funcionalidad.

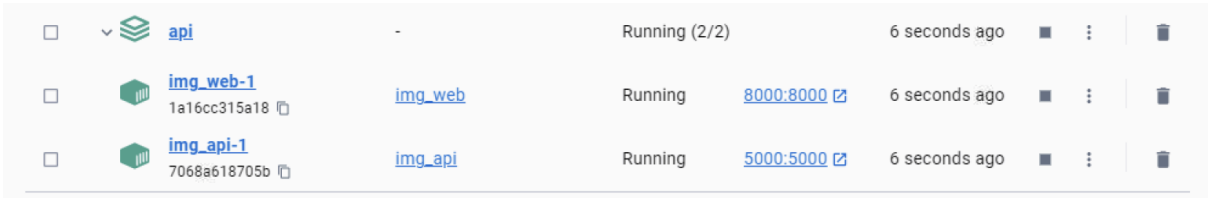


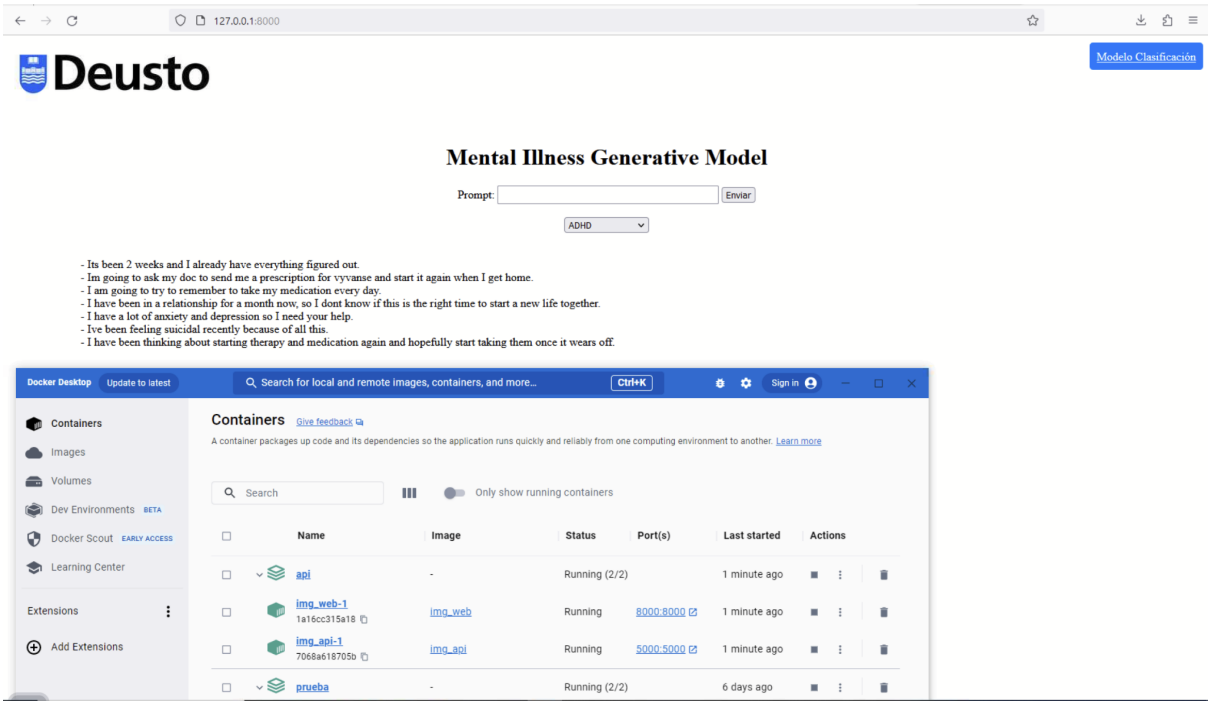
Para desplegar el proyecto de forma local, se debe ejecutar por separado la API y la WEB, tal y como se muestra en la siguiente imagen. Este método de despliegue es el que se ha usado para hacer las pruebas a lo largo del desarrollo del proyecto.



La opción de ejecutarlo de forma local es la más sencilla y rápida, tanto en términos de tiempo de ejecución del modelo como en tiempo de despliegue (siempre y cuando se tenga el entorno virtual preparado). Cabe destacar que la ejecución del modelo sin GPU supone un tiempo de espera de unos 10 segundos desde que se solicita la ejecución del modelo (desde que se introduce un prompt).

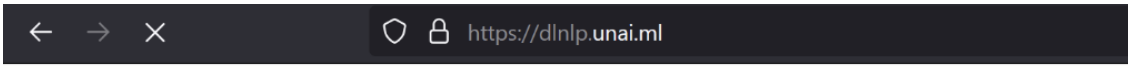
Adicionalmente, se ha configurado el código fuente de forma que se pueda Dockerizar de manera opcional.





Esta solución es rápida en el despliegue y extremadamente cómoda para compartir. En el lado negativo, está la subida de latencia respecto al método anterior.

Por último, hemos decidido subir la solución dockerizada a un servidor en el dominio público. Concretamente, la solución se puede encontrar temporalmente en: https://dlnlp.unai.ml/. Esta última opción de despliegue, a pesar de parecer muy sencilla partiendo ya del proyecto en docker, requiere una configuración muy laboriosa en el servidor. Adicionalmente, la latencia alcanza el minuto.



Deusto

Mental Illness C

Prompt:

4.- CÓDIGO

1. Todo el código y los resultados obtenidos en el proyecto está almacenado en drive. Puede accederse a dicho contenido desde el siguiente enlace: drive
2. También está el código disponible en el siguiente repositorio de github: github

5.- RESULTADOS

5.- DESVIACIONES TÉCNICAS

1. No hemos usado el dataset de la Universidad de Georgetown. Se ha decidido no insistir en la firma de la documentación con el fin de evitar problemas legales. En cambio, se ha utilizado un dataset alternativo, de menor calidad, obtenido de zenodo. La fuente de la que se han capturado datos es la misma: reddit. Se han utilizado todos los archivos excepto las menos relacionadas con la salud mental o los que pueden contaminar el proceso de aprendizaje (divorcio, …). La calidad del dataset se ha visto enormemente deteriorada ya que el propio de la UG no tiene registros que hablan de forma explícita de la enfermedad en cuestión.

2. Por el mismo motivo, se descarta la opción de usar la herramienta en sesiones de terapia, ya que el modelo no se ajusta al caso de uso.

3. En cuanto a los propósitos de la implementación del modelo generativo, podría decirse que en primera instancia eran dos: Por un lado, se buscaba desarrollar una red antagonista para reforzar el aprendizaje del modelo, y, por otro, se pretendía representar en forma de texto la manera en la que una persona con la condición mental seleccionada respondería al prompt de entrada. Tras entrenar el modelo generativo, se ha podido observar que los textos generados por este son demasiado explícitos como para considerarse de calidad y retroalimentar el modelo base. Por este motivo, no se ha desarrollado dicha retroalimentación ya que se considera que existe riesgo de sobreentrenar el modelo, dejando fuera de alcance el primero de los propósitos marcados al comienzo del planteamiento del proyecto, ya que la generación de textos de baja calidad podría afectar negativamente al modelo base.

4. Finalmente se ha descartado el desarrollo del módulo de speech to text propuesto en la entrega parcial del proyecto, al considerar que no aporta suficiente valor a las tareas de clasificación y generación, y teniendo en cuenta las limitaciones temporales para la entrega de este.

5. Por otra parte, se ha optado por añadir una función de generación de imágenes a través de prompts y un modelo pre-entrenado de stable-diffusion, el cual permite ilustrar de una forma gráfica un concepto que represente cada una de las enfermedades para las que se han entrenado los modelos de generación de texto.

6. Tratar de integrar el proyecto en una solución visual para prestar el servicio a cualquier usuario no experto, ha supuesto un tiempo mayor a lo esperado por todos los problemas técnicos que han ido surgiendo a lo largo del desarrollo. Estos contratiempos se han pagado con una menor calidad en el frontend y brechas de seguridad en el servidor.

6.- APORTACIONES DE CADA INTEGRANTE

Antes de comenzar con las aportaciones de cada integrante, es importante indicar que el primer paso, el cual ha sido realizado de forma conjunta por parte de todos los integrantes del grupo, se ha basado en obtener el dataset señalado en la documentación. Posteriormente, se ha generado un script en el que se ha obtenido un fichero *.csv el cual contiene de forma combinada todos los registros disponibles para cada una de las condiciones mentales.

Comenzando con las aportaciones de cada integrante, en primer lugar, cabe señalar que Iker ha sido la persona encargada de realizar la preparación de los datos empleados en el posterior entrenamiento de los modelos encargados de realizar la clasificación de un prompt en función de una cierta condición mental, así como de realizar el posterior entrenamiento de dichos modelos.

Para ello, se ha comenzado con la limpieza de los datos. Para dicha limpieza, como el objetivo de este apartado es la clasificación de texto, se ha optado por un preprocesado bastante estándar. Para empezar, se ha pasado todo el texto a minúsculas y se han eliminado los carácteres especiales además de los emoticonos y las urls. Por otra parte, teniendo en cuenta que los textos están en inglés, se han eliminado las stopwords que aportan poco valor a la tarea de clasificación. Por último, es importante conocer el origen de los datos, los cuales vienen de diversos subreddits en los que se utilizan diversas abreviaciones y términos específicos de la red social (tdlr, cont, adv, mod…). Para lidiar con esta terminología, se han sustituido/eliminado varios textos en los que aparecían, dependiendo de si aportan información posiblemente relevante (asap = as soon as possible) o no (cont = continuación del texto anterior).

Una vez obtenidos los datos limpios y listos para ser utilizados, se ha optado por probar los 2 modelos a utilizar, siendo estos BERT y BigBird, para una clasificación binaria. Esto se ha hecho para poder seleccionar el modelo con mejores resultados y entrenarlo para la clasificación de todas las condiciones mentales tratadas, con el objetivo de reducir los costes de computación y de obtener los mejores resultados posibles en el poco tiempo disponible.

Para el tokenizado de los datos, se han utilizado tokenizadores preentrenados de ambos modelos, tratando de obtener los mejores resultados posibles con cada uno de ellos.

En lo referente a las pruebas de clasificación binaria, se ha optado por tratar de clasificar los textos pertenecientes a usuarios con ansiedad y ADHD. Para ello, se han cargado los datos en 2 notebooks, se ha realizado un preprocesado de texto pensado para la clasificación, se han cargado los modelos BERT y el BigBird en los notebooks, y se ha procedido a reentrenar dichos modelos con los datos preprocesados. Una vez entrenados ambos modelos, se ha podido comprobar que, aunque no por un gran margen, el modelo BigBird mostraba mejores resultados a la hora de clasificar los textos. Como consecuencia de esta prueba, se ha tomado la decisión de seleccionar el modelo BigBird como modelo de clasificación final a la hora de tratar de entrenar un modelo para todas las condiciones tratadas.

Para el modelo de clasificación general, se ha seguido un proceso muy similar al de los clasificadores binarios. Para empezar se han cargado los datos y se ha realizado un preprocesado del texto en el que se han eliminado emojis, urls, carácteres especiales y stopwords, además de pasar todo el texto a minúsculas. Una vez preprocesado el texto, se ha pasado a la tokenización con el BigBirdTokenizer. Por último, se ha cargado un modelo preentrenado de BigBird, el cual ha sido actualizado con los textos tokenizados, logrando de esta forma un modelo de clasificación de 9 posibles clases, con una precisión de cerca del 70%.

Por otra parte, Lander ha sido la persona encargada de llevar a cabo la preparación de los datos empleadas en el posterior entrenamiento de los modelos generativos, así como de realizar el entrenamiento de estos.

Para ello, en primer lugar, se ha generado un script que permite generar un corpus específico para cada una de las condiciones mentales deseadas. En dicho script, se ha importado el fichero *.csv combinado generado al inicio del desarrollo del proyecto. Posteriormente, se ha procedido a realizar una limpieza de los textos presentes en los registros del dataset importado, mediante la definición de ciertas expresiones regulares para la eliminación de emoticonos, símbolos, URLs, y similares. Una vez realizada la limpieza de los textos contenidos en los registros, se ha empleado un filtro a fin de obtener un corpus específico para cada una de las enfermedades mentales, los cuales han sido almacenados en formato *.txt.

Tras la generación de los corpus, se ha generado un script de entrenamiento para cada una de las condiciones mentales para las que se deseaba entrenar un modelo generativo. En cada una de ellas se ha procedido a la carga del corpus específico generado previamente, y se ha elaborado una lista que contiene los documentos divididos, teniendo en cuenta los saltos de línea. A continuación se ha realizado la creación del script del dataset utilizando la librería de Hugging Face. Para ello, se genera una lista de diccionarios con los campos “id”; “text” para cada uno de los documentos segmentados previamente. Posteriormente, se crea el dataset con la lista de diccionarios, y se realiza el split entre train, validation y test data. Dichos ficheros de texto son almacenados para su posterior uso en el entrenamiento del modelo. A continuación, se realiza el entrenamiento del modelo utilizando el script run_clm.py. Para ello se instala la librería transformers, y se clona de Hugging Face la herramienta transformers.git. Se definen los parámetros de configuración del modelo, como puede ser el learning rate, steps, epochs, etc. Y posteriormente se ejecuta el comando para realizar el entrenamiento, una vez definidas las rutas de salida de output y logs, donde se almacenarán los checkpoints de los modelos que se vayan entrenando y los logs, respectivamente.

Con ello se da por finalizado el entrenamiento de los modelos generativos. Se debe tener en cuenta que el proceso descrito se ha llevado a cabo para cada una de las condiciones mentales a partir de las cuales se entrenaría un modelo generativo.

A continuación, una vez realizado el entrenamiento de los modelos generativos, entre todo el equipo se ha procedido a realizar una investigación sobre los modelos generativos de imágenes. Tras llevar a cabo lo mencionado, se ha concluido que el modelo más eficiente y

el que mejor se ajustaba a las necesidades identificadas, se trataba de “Stable Diffusion”. Con ello, se ha procedido a generar diversas imágenes de forma secuencial, variando el valor del parámetro de semilla para cada imágen generada, así como el prompt de generación para cada conjunto de imágenes generadas para cada condición mental incluida, obteniendo un resultado final de un conjunto de imágenes relacionadas con cada una de las condiciones mencionadas.

Por último, una vez realizados los entrenamientos de los diversos modelos planteados, tanto para cubrir la funcionalidad de generativo descrita, como para cubrir la funcionalidad de clasificación mencionada, así como la generación de imágenes asociadas a ciertas condiciones mentales, Unai ha sido el encargado de llevar a cabo la unificación de cada uno de los módulos desarrollados y descritos previamente, así como de realizar una integración de estos en un servicio de API, a fin de que estos puedan ser consumidos de una forma mucho más vistosa.

Para ello, el primer paso ha sido generar un código que únicamente importe las librerías necesarias para operar (eliminando todas las referentes a la limpieza de datos y demás), además de que cargue los modelos ya entrenados y los ejecute admitiendo unos parámetros de entrada provenientes del usuario. Toda esta funcionalidad viene recogida en un archivo .py dedicado a ello, definida en distintas funciones para hacer más legible el código.

Después, se ha creado la API Flask de forma incremental, es decir, se han implementado las funcionalidades básicas en un principio y después se ha ido enriqueciendo a medida que se han añadido otras características. Se ha probado la API mediante programas externos (Postman y requests.py).

Una vez creado el backend se ha procedido a diseñar e integrar el frontend. Para el frontend también se ha usado Flask por su simpleza. Teniendo ya el backend operativo, se han incluido las peticiones a la API en el frontend, logrando integrar ambas partes del proyecto a desplegar.

Adicionalmente, el miembro también ha estado presente en el resto de partes del proyecto, prestando ayuda en el resto de fases.

Finalmente, todo el equipo ha sido partícipe de la elaboración de la presente documentación, así como de la presentación mediante la cual se realiza la exposición de la herramienta desarrollada.