Ignacio Gimeno Alonso

PROYECTO FIN DE BOOTCAMP

Desarrollo de un enrutador de preguntas sobre enfermedades

Introducción

La integración de la inteligencia artificial (IA) en el ámbito de la salud ha experimentado un crecimiento significativo en los últimos años, impulsada por la necesidad de mejorar el acceso a la información médica y la eficiencia en la atención al paciente. Una de las aplicaciones más prometedoras de la IA en este campo es el desarrollo de chatbots, sistemas automatizados capaces de interactuar con los usuarios mediante lenguaje natural. Estos chatbots pueden desempeñar un papel crucial en la prestación de servicios de información médica, al proporcionar respuestas rápidas y precisas a las preguntas de los pacientes, al tiempo que alivian la carga de trabajo del personal sanitario.

En este trabajo, presentamos el desarrollo de un chatbot especializado en responder preguntas médicas. Este chatbot ha sido diseñado para asistir a los usuarios proporcionando información confiable y basada en evidencia sobre diversas condiciones de salud, tratamientos, y recomendaciones preventivas. La implementación de sistemas como este podría abordar varias problemáticas clave en el sector de la salud, tales como la limitada disponibilidad de profesionales médicos, las largas esperas para obtener consultas, y la dificultad de acceso a información médica actualizada y comprensible.

El chatbot se basa en tecnologías de procesamiento de lenguaje natural (NLP) e inteligencia artificial, lo que le permite comprender y responder de manera efectiva a una amplia gama de preguntas médicas.

A lo largo de este trabajo, describiremos el proceso de desarrollo del chatbot, incluyendo el diseño de su arquitectura, el entrenamiento del modelo mediante los datos médicos de los que disponemos, y la implementación de los algoritmos empleados para el desarrollo de la solución. Finalmente, se comentarán los resultados de las distintas métricas planteadas y se propondrán mejoras del modelo.

Desarrollo de la solución

Acceso a datos e identificación de objetivos

Los datos de los que disponemos para realizar el entrenamiento del modelo son los presentes en el siguiente repositorio: *https://github.com/abachaa/LiveQA_MedicalTask_TREC2017/tree/master*.
Consta de dos archivos para el entrenamiento del modelo y uno para el test. Cada archivo está formado por pares de preguntas y respuestas médicas, y en cada pregunta está indicado el tipo de pregunta (type), el tema (focus) del que trata y un título (subject) opcional. Todos estos archivos están en formato .XML, por lo que hemos empleado la biblioteca xml, en concreto el módulo

xml.etree.ElementTree para realizar la lectura de los archivos y la separación de los distintos campos presentes.

En el primer paso de nuestra solución intentaremos identificar el tipo de pregunta, para lo cual nos fijaremos en la variable ‘type’. En el segundo paso, buscaremos las palabras importantes de la pregunta, es decir, intentaremos predecir la variable ‘focus’. Por último, responderemos a la pregunta planteada, empleando la variable ‘answer’.

Diseño del flujo de trabajo

El desarrollo de la solución constará de las siguientes etapas:

1. Preprocesado de los datos de partida para las distintas fases de desarrollo.
2. Entrenamiento y evaluación de un modelo para la clasificación del tipo de pregunta (‘type’).
3. Fine-tuning y evaluación de un modelo preentrenado para la extracción de las palabras clave (‘focus’) de las preguntas.
4. Fine-tuning de un modelo preentrenado para la elaboración de las respuestas a las distintas preguntas.

Análisis de los datos de partida

Los datos de partida constan de los siguientes elementos por cada pregunta (‘MESSAGE’): un título opcional que explica el tema de la pregunta (‘SUBJECT’); las distintas subpreguntas (‘SUB-QUESTION’) en las que se divide la pregunta principal; el tipo (‘TYPE’) y tema principal (‘FOCUS’) de cada subpregunta; y una o varias respuestas (‘ANSWER’) a cada subpregunta.

Mediante un análisis preliminar de los datos de partida, podemos observar que en el conjunto de los dos archivos de entrenamiento tenemos 23 tipos distintos de preguntas, siendo el predominante ‘treatment’ con 195 preguntas, seguido por ‘information’ con 73 preguntas. En el otro extremo, tenemos tipos como ‘diagnose_me’, ‘genetic changes’ o ‘ingredient’ que solo disponen de una pregunta cada uno (Figura 1). Debemos tener en cuenta que, como se explicará en el apartado de preprocesado de los datos, nos hemos quedado con un único tipo por pregunta para simplificar el desarrollo de la solución, aunque haya preguntas con varios tipos.

Respecto al ‘focus’ de las preguntas de entrenamiento, tenemos más de 300 temas distintos, siendo el tema más presente el lupus, con 5 preguntas (Figura 2).

Respecto al conjunto de test, tras quedarnos con un único tipo por pregunta y realizar un preprocesado con el cambio de algunos tipos, disponemos finalmente de 20 tipos de preguntas, siendo el mayoritario ‘information’ con 21 preguntas, mientras que ‘prevention’ o ‘inheritance’ contienen una única pregunta. Los tipos ‘association’, ‘genetic changes’ y ‘diagnose_me’ no están contemplados en este conjunto.

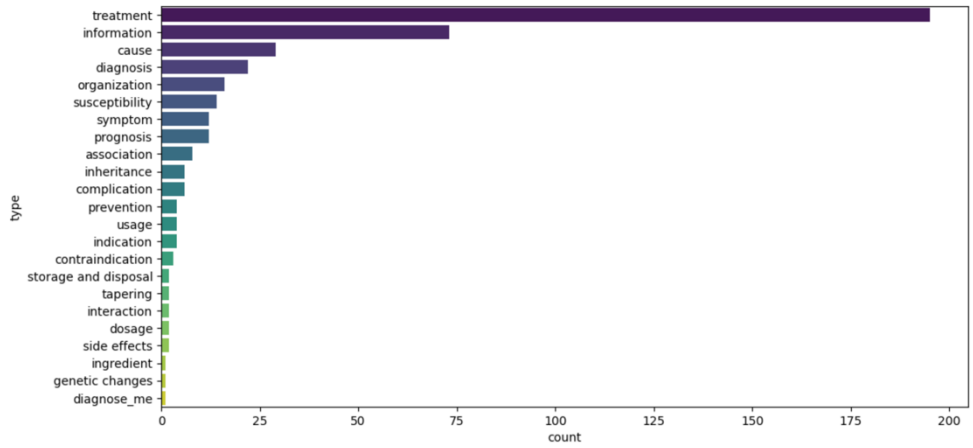

Figura 1: Cantidad de preguntas por tipo en el conjunto de entrenamiento.

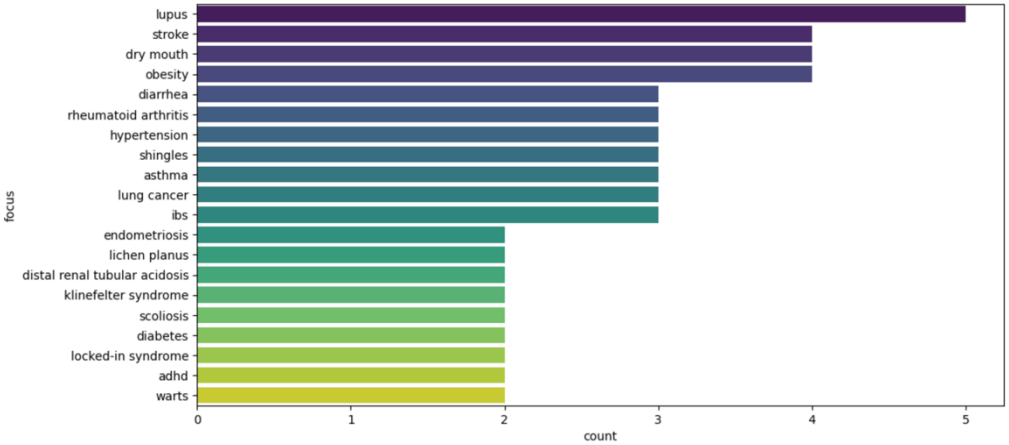

Figura 2: Cantidad de preguntas por tema en el conjunto de entrenamiento.

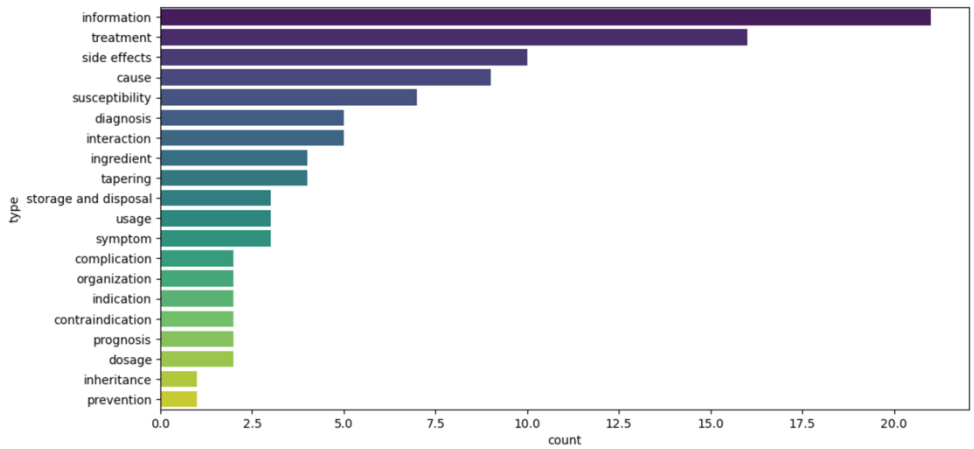

Figura 3: Cantidad de preguntas por tipo en el conjunto de test.

Selección de métricas

Para comprobar el desempeño del modelo estudiaremos distintas métricas, tanto de evaluación de los distintos componentes de la solución como técnicas de la inferencia. Por un lado, nos fijaremos en la precisión de los modelos empleados para la clasificación de las preguntas, y para la extracción del tema principal de

cada pregunta. Por otro lado, podemos fijarnos en el tiempo transcurrido desde que se le pasa una pregunta a la solución hasta que nos da una respuesta, así como en la calidad de la respuesta. Para evaluar este último punto, implementaremos un feedback loop que explicaremos en el apartado correspondiente.

Preprocesamiento de datos

En primer lugar, al leer ambos archivos de entrenamiento hemos pasado todo el texto que hemos guardado (preguntas, respuestas, tipo y focus) a minúsculas. Respecto a la limpieza de las preguntas, tras pasarlas a minúsculas hemos eliminado las etiquetas HTML en caso de que alguna se hubiera quedado en el texto, hemos sustituido o eliminado los caracteres HTML como &, >, o " entre otros, hemos eliminado posteriormente todos los caracteres no alfanuméricos y finalmente hemos eliminado las stopwords. De esta forma hemos creado un dataframe con dos columnas de preguntas: una con las preguntas completas en minúsculas y sin caracteres especiales, y otra en la que además hemos eliminado las stopwords. La primera columna de preguntas nos servirá para refinar los modelos que emplearemos para detectar el tema de las preguntas y para elaborar las respuestas, mientras que las preguntas con las stopwords eliminadas las emplearemos para entrenar el modelo de detección del tipo de pregunta.

Como ya hemos comentado anteriormente, para simplificar el desarrollo de la solución hemos decidido que cada pregunta sea de un único tipo, eliminando el resto de los tipos, si los hubiera, de la variable. De esta forma hemos obtenido 23 tipos distintos de preguntas, habiendo 15 tipos con menos de 10 preguntas cada uno, mientras que el tipo ‘treatment’ presenta casi 200 preguntas e ‘information’ contiene 73 preguntas. Debido a alto desbalanceamiento presente, hemos decidido reducir los tipos de preguntas a 9. Esto está motivado por el hecho de que hay tipos de pregunta que se pueden solapar entre sí, como los asociados a medicamentos, la mayoría de los cuales se pueden agrupar en ‘information’. Mediante una revisión de las preguntas asociadas a los distintos tipos, hemos decidido dejar los siguientes 9 tipos de pregunta: ‘treatment’, ‘information’, ‘susceptibility’, ‘prognosis’, symptom’, ‘diagnosis’, ‘cause’, ‘organization’ y ‘drug interaction’. De esta forma nos aseguramos un mejor rendimiento del modelo que emplearemos para la clasificación de las preguntas.

Tras el proceso de preprocesado de los datos, hemos obtenido finalmente dos dataframes distintos, uno de entrenamiento y otro de test. Ambos dataframes tienen 6 columnas: id, pregunta, pregunta preprocesada, tema (focus), tipo de pregunta, y respuesta. La respuesta no ha sido preprocesada ya que suponemos que están escritas en un formato adecuado para su empleo en el refinamiento del modelo que genere las respuestas finales. Para el entrenamiento de cada parte de la solución hemos empleado unas columnas distintas: para el entrenamiento del modelo de clasificación del tipo de pregunta hemos empleado las columnas de la pregunta preprocesada y el tipo de pregunta. Para el refinamiento del modelo de extracción del tema de la pregunta hemos empleado las columnas de la pregunta completa y la del tema. Finalmente, para el refinamiento del modelo empleado para elaborar las respuestas hemos

empleado una nueva columna con las preguntas reformuladas a partir de su tipo y su tema, y la columna de respuestas.

Arquitectura de modelo

La solución propuesta consta de tres componentes principales. El primero es un clasificador mediante bosques aleatorios que detecta el tipo de pregunta que tenemos. Hemos decidido emplear un clasificador de bosques aleatorios tras comparar su rendimiento con el de distintas redes neuronales creadas para este fin. Para poder entrenar este modelo, primero hemos entrenado y guardado un modelo Word2Vec [1] para generar los embeddings y pasarle al algoritmo de bosques aleatorios vectores numéricos. Una vez hecho esto, hemos generado dos conjuntos distintos de entrenamiento: uno con el total de los datos de entrenamiento, y otro con un submuestreo de los datos más representados para conseguir un conjunto balanceado. Tras obtener métricas similares con el modelo entrenado con ambos conjuntos de datos, hemos decidido guardar para su uso posterior el modelo entrenado con los datos balanceados, ya que será capaz de distinguir mejor los distintos tipos de preguntas.

En segundo lugar, hemos empleado el modelo preentrenado de extracción de palabras clave de VoiceLab vlt5-base-keywords [2]. Este modelo está basado en el modelo T5 de Google [3], y es capaz de extraer las palabras clave de un texto. Hemos realizado un refinamiento del modelo mediante nuestros datos de entrenamiento, pasándole al modelo las preguntas completas y el campo ‘focus’.

Finalmente, para la elaboración de las respuestas hemos empleado el modelo distilgpt2 de distilbert [4]. Lo hemos refinado empleando nuestros pares de pregunta-respuesta, empleando las preguntas completas. La elección de este modelo frente a otros más grandes como GPT-2, GPT-Neo 2.7B o Meta Llama 3.1 es su tamaño, ya que con esta elección hemos sido capaces de cargar el modelo junto con el modelo vlt5-base-keywords tanto en Google Colab como en nuestro ordenador personal sin problemas de memoria. Para refinar este modelo le hemos pasado las preguntas reformuladas a partir de su tipo real y su focus, junto con la respuesta dada en el dataframe inicial de entrenamiento.

Evaluación del modelo

Para realizar la evaluación del modelo hemos recopilado dos tipos distintos de métricas. Por un lado, en cada inferencia del modelo guardamos y mostramos el tiempo transcurrido en cada paso (tiempo de clasificación de la pregunta, tiempo de extracción del tema, y tiempo de elaboración de la respuesta). En nuestro ordenador el tiempo transcurrido desde que se hace la pregunta hasta que el modelo muestra la respuesta ronda los 5 segundos totales, algo que puede mejorar en sistemas con mejor equipamiento.

Por otro lado, hemos estudiado métricas de precisión de los distintos componentes de la solución. Hay dos partes del modelo de las cuales podemos realizar una evaluación inmediata de precisión y desempeño: la clasificación del tipo de pregunta, y la extracción del tema. Para la clasificación del tipo de pregunta hemos calculado las métricas de evaluación de la clasificación

mediante el modelo de bosques aleatorios. Con el conjunto de entrenamiento balanceado, hemos obtenido una accuracy del 24%, precisión promediada del 40%, recall promediado del 24% y F1-score promediado del 26%. Nuestro modelo detecta preguntas de todos los tipos, mientras que al emplear el modelo entrenado con los datos sin balancear hay tipos de preguntas del conjunto de test que no se llegan a detectar.

El motivo de obtener unas métricas tan bajas puede residir en la poca cantidad de ejemplos de cada tipo distinto de pregunta que hemos empleado para entrenar, ya que al balancear el conjunto de entrenamiento nos hemos quedado con únicamente 12 ejemplos de cada uno de los 9 tipos de pregunta que hemos considerado. Una forma de mejorar el rendimiento de esta parte de la solución consistiría en aumentar la cantidad de ejemplos de pregunta por tipo. Además, también podría revisarse la redistribución que hemos realizado en 9 tipos de preguntas para intentar mejorarla.

Respecto a la extracción del tema de las preguntas, tras refinar el modelo empleado hemos calculado la cantidad de temas predichos que coinciden con el focus dado para cada pregunta. Esta métrica nos indica que el 36.5% de los temas predichos encaja con el focus de las preguntas. Respecto a los temas que no coinciden, hemos comprobado que algunos temas coinciden, pero se añade alguna palabra que no aparece en el focus real (por ejemplo, se predice el tema ‘Lyme Disease symptoms’ mientras que el tema real es ‘Lyme Disease’). Estos casos los contamos como error, aunque a la hora de reformular la pregunta para pasársela al modelo que elabora las respuestas es posible que el desempeño sea correcto.

Finalmente, en el modelo de elaboración de las respuestas no podemos obtener métricas de precisión como con los modelos anteriores. En este caso, debe ser el usuario el que indique si la respuesta dada es adecuada a su pregunta o no. Para obtener esta métrica, hemos incluido un feedback loop en el cual se le pregunta al usuario si la respuesta es adecuada en el 10% de las respuestas.

Desarrollo de los componentes de inferencia/producción

Diseño del flujo de trabajo

Una vez hemos diseñado la solución final con los modelos que vamos a emplear, hemos diseñado el flujo de trabajo que seguirá nuestro programa. Este flujo consta de seis partes principales:

1. **Carga de los modelos**: en primer lugar, realizamos la carga de los cuatro modelos que emplea la solución (Word2Vec, bosques aleatorios para la clasificación por tipo de pregunta, vlt5-base-keywords refinado para la extracción del tema de la pregunta, y distilgpt2 para la elaboración de la respuesta).
2. **Realización de la pregunta por el usuario**: se le pide al usuario que haga la pregunta, y esta se guarda en una variable.

3. **Clasificación de la pregunta**: el siguiente paso consiste en realizar el preprocesado de la pregunta realizada por el usuario y la clasificación por tipo de pregunta.
4. **Extracción del tema**: en el cuarto paso preprocesamos la pregunta para extraer el tema y se la pasamos al modelo encargado de ello, obteniendo la predicción del tema de la pregunta.
5. **Elaboración de la respuesta**: en este paso elaboramos la pregunta reformulada a partir del tipo de pregunta y su tema y se la pasamos al modelo encargado de elaborar la respuesta.
6. **Feedback loop**: por último, se ha incorporado un feedback loop que pregunte al usuario en el 10% de las respuestas dadas si esta encaja con lo preguntado.

Diseño y definición de los contratos de datos

Considerando el tipo de solución que tenemos (un chatbot de preguntas y respuestas médicas), los datos de entrada al modelo podrían consistir en un diccionario con el identificador del usuario, el timestamp, y la pregunta planteada, todo en texto plano con codificación UTF-8. También se podría incluir, de forma opcional y tras recibir el consentimiento del usuario, su edad y género para mejorar las respuestas.

La salida consistiría en otro diccionario con el identificador de la respuesta, el timestamp y la respuesta dada, de nuevo en texto plano con codificación UTF-8.

Respecto a las políticas de acceso y uso de los datos, únicamente personal autorizado y sistemas con permisos específicos podrían acceder a los datos, ya que aparecería información personal, y solo se podrían emplear para mejorar las respuestas del chatbot y analizar su desempeño.

Implementación de funciones para la preparación de los datos

Nuestra solución emplea distintas funciones para preparar los datos según el paso en el que se encuentre. Para clasificar la pregunta según su tipo, empleamos dos funciones de preparado de datos: preprocess_question_type, que preprocesa el texto de la pregunta y elimina las stopwords, y vectorize_question, que transforma la pregunta a un vector numérico mediante un modelo Word2Vec.

A la hora de detectar el tema de la pregunta, el preprocesado de la pregunta se hace mediante la función preprocess_question_focus, que prepara el prompt que se le pasa al modelo de extracción del tema de la pregunta y lo tokeniza empleando el tokenizador del modelo.

A la hora de preparar la respuesta a la pregunta, empleamos la función generate_QA_question, que genera la pregunta reformulada a partir del tipo y el tema extraídos, y posteriormente generamos el prompt para el modelo y lo tokenizamos con el tokenizador propio del modelo de generación de texto.

Implementación de funciones para la inferencia de los modelos

Para realizar la inferencia del modelo, en la solución hemos implementado tres funciones: detection_type, que clasifica la función según su tipo mediante el clasificador de bosques aleatorios; extract_focus, que detecta el tema de la pregunta mediante el modelo vlt5-base-keywords refinado; y answer_question, que genera la respuesta a la pregunta reformulada mediante el modelo distilgpt2 refinado con nuestros datos. En esta última función, empleamos también la función clean_answer para realizar un postprocesado de la respuesta y que se muestre correctamente.

La ejecución de la solución se hace mediante la consola de comandos, ejecutando la orden *python main.py*. Al hacerlo, se nos pedirá escribir la pregunta y posteriormente se mostrará la respuesta junto con los tiempos de ejecución de los distintos pasos. Para dejar de hacerle preguntas al chatbot y paralizar su ejecución debemos pulsar ctrl+c.

Implementación de funciones para la monitorización del modelo

Para realizar la monitorización de la solución tenemos cuatro variables que nos indican el tiempo transcurrido en cada paso de la solución y el tiempo total. La monitorización de estas variables durante la permanencia del modelo en producción nos puede servir para comprobar si ocurre algo raro en alguno de los pasos de la solución. También tenemos una variable, obtenida mediante el feedback loop que explicaremos a continuación, que nos indica si la respuesta dada es adecuada. Mediante la aplicación de funciones para la monitorización de estas variables podemos realizar un seguimiento del desempeño de la solución y valorar cuándo es necesario realizar un reentrenamiento de los distintos modelos involucrados.

Diseño e implementación del feedback loop

La solución preparada en el presente trabajo da respuesta a las preguntas médicas que se le plantean. Debido a su naturaleza, la precisión de las respuestas solo puede valorarse mediante la opinión del usuario. Por ello, hemos incorporado un feedback loop que pregunta en un 10% de las respuestas si esta es adecuada a lo que se le ha preguntado. Esto se ha realizado mediante la generación de un número aleatorio del 1 al 10 y la realización de esta pregunta si el número generado es 1. Se pide que la respuesta a esta pregunta sea ‘y’ (yes, respuesta adecuada) o ‘n’ (no, respuesta no adecuada). Mediante el almacenamiento y la monitorización de esta variable podríamos valorar cuándo es necesario realizar una mejora o un reentrenamiento de la solución.

Conclusiones y trabajo futuro

En este trabajo hemos conseguido entrenar y desarrollar un chatbot capaz de responder a preguntas médicas. Para ello hemos realizado un entrenamiento de

un clasificador de bosques aleatorios para clasificar el tipo de pregunta, un refinamiento del modelo vlt5-base-keywords para detectar el tema de la pregunta y un refinamiento del modelo distilgpt2 para elaborar la respuesta a la pregunta. Las métricas que hemos obtenido (precisión del clasificador, precisión del extractor del tema, tiempo de inferencia) son bastante pobres, obteniendo un 24% de precisión en el clasificador, un 37.5% en el extractor de temas y un tiempo de inferencia en torno a 5 segundos. Respecto a la calidad de las respuestas dadas, tras realizar varias pruebas hemos comprobado que el modelo suele contestar mal, respondiendo a cosas que no se han preguntado o inventándose la información. Para conseguir una mejora del rendimiento del chatbot, deberíamos conseguir una mayor cantidad de datos de preguntas y respuestas clasificadas según los 9 tipos de pregunta empleados durante el entrenamiento, así como emplear un modelo mejor y más grande para la elaboración de la respuesta. Sin embargo, para el empleo de un modelo de generación de texto más grande sería necesario emplear un equipamiento mejor que el empleado para la realización del trabajo, ya que se necesitaría una memoria de CPU y GPU mayor que de la que se disponía.

Respecto a la entrada en producción de la solución, en el futuro sería necesario desarrollar una API a través de la cual conectarse con el modelo. Esta API debería ser capaz de almacenar las distintas métricas generadas para poder realizar una correcta monitorización de la solución.

Bibliografía

[1] T. Mikolov, K. Chen, G. Corrado y J. Dean, «Efficient Estimation of Word Representations in Vector Space,» *arXiv,* vol. 1301.3781, 2013.

[2] Hugging Face, «vlt5-base-keywords,» [En línea]. Disponible en: https://huggingface.co/Voicelab/vlt5-base-keywords.

[3] Hugging Face, «T5 Base,» [En línea]. Disponible en: https://huggingface.co/google-t5/t5-base.

[4] Hugging Face, «DistilGPT2,» [En línea]. Disponible en: https://huggingface.co/distilbert/distilgpt2.