Universidad Nacional de Loja

Facultad de la Energía, las Industrias y los Recursos Naturales No Renovables

Carrera de Computación

Modelo QA basado en DistilBERT para responder a preguntas sobre el contenido extraído de tareas académicas de la carrera de Computación de la UNL.

QA model based on DistilBERT to answer questions about content extracted from academic assignments of the Computer Science course at UNL.

Trabajo de Integración Curricular previa a la obtención del título de Ingeniero en Ciencias de la Computación.

AUTOR:

Edy Francisco Jiménez Merino

DIRECTOR:

Ing. Oscar Miguel Cumbicus Pineda, Mg.Sc

Loja – Ecuador

2024

Certificación

Loja, 20 de septiembre del 2024

Oscar Miguel Cumbicus Pineda, Mg.Sc.
DIRECTOR DEL TRABAJO DE TITULACIÓN

CERTIFICO:

Que he revisado y orientado todo el proceso de elaboración del Trabajo de Integración Curricular denominando: **Modelo QA basado en DistilBERT para responder a preguntas sobre el contenido extraído de tareas académicas de la carrera de Computación de la UNL**, previo a la obtención del título de Ingeniero en Ciencias de la Computación, de la autoría del estudiante Edy Francisco Jiménez Merino, con **cédula de identidad Nro**. 1950170389, una vez que el trabajo cumple con todos los requisitos exigidos por la Universidad Nacional de Loja, para el efecto, autorizo la presentación del mismo para su respectiva sustentación y defensa.

Ing. Oscar Miguel Cumbicus Pineda, Mg.Sc.
DIRECTOR DEL TRABAJO DE TITULACIÓN

Autoría

Yo, **Edy Francisco Jiménez Merino**, declaro ser autor del presente Trabajo de Integración Curricular y eximo expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos, de posibles reclamos y acciones legales, por el contenido del mismo. Adicionalmente acepto y autorizo a la Universidad Nacional de Loja la publicación de mi Trabajo de Integración Curricular, en el Repositorio Digital Institucional –Biblioteca Virtual.

Firma: 

Cédula de identidad: 1950170389
Fecha: 20 de septiembre del 2024
Correo electrónico: edy.jimenez@unl.edu.ec
Teléfono: +593 939943013

Carta de autorización por parte del autor, para consulta, reproducción parcial o total y/o publicación electrónica del texto completo, del Trabajo de Integración Curricular

Yo, **Edy Francisco Jiménez Merino**, declaro ser el autor del Trabajo de Integración Curricular denominado: **Modelo QA basado en DistilBERT para responder a preguntas sobre el contenido extraído de tareas académicas de la carrera de Computación de la UNL**, autorizo al sistema Bibliotecario de la Universidad Nacional de Loja para que, con fines académicos, muestre la producción intelectual de la Universidad, a través de la visibilidad de su contenido en el Repositorio Institucional.

Los usuarios pueden consultar el contenido de este trabajo en el Repositorio Institucional, en las redes de información del país y del exterior con las cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia del Trabajo de Integración Curricular que realice un tercero.

Para constancia de esta autorización, suscribo, en la ciudad de Loja, a los veinte días del mes de septiembre de dos mil veinticuatro.

Firma: 

Autor: Edy Francisco Jiménez Merino
Cédula de identidad: 1950170389
Dirección: Loja (Inés Jimenez - Atahualpa)
Correo electrónico: edy.jimenez@unl.edu.ec
Teléfono: +593 939943013

DATOS COMPLEMENTARIOS:

Director del Trabajo de Integración Curricular: Ing. Óscar Miguel Cumbicus Pineda, Mg.Sc.

Dedicatoria

A mis queridos padres, por su apoyo constante que han sido la fuerza impulsora para poder estudiar en la Universidad nacional de Loja. A mis hermanos, que con su compañía han hecho este viaje más llevadero. Gracias por ser mi inspiración y por creer en mí incluso en los momentos más difíciles. Esta investigación es para ustedes.

Edy Francisco Jiménez Merino

Agradecimiento

Deseo expresar mi más sincera gratitud a quienes han sido fundamentales en este recorrido. Agradezco especialmente al Ing. Oscar Miguel Cumbicus Pineda Mg.Sc, cuya guía y dedicación han sido invaluables para el desarrollo y éxito de este trabajo de integración curricular. Mi profundo reconocimiento se extiende a mis padres, pilares inquebrantables, cuyo apoyo incondicional han sido mi mayor fortaleza, y a mis hermanos, por su constante aliento y compañía en este desafío. Este logro es, sin duda, un reflejo del respaldo de todos aquellos que, de diversas maneras, han contribuido a mi formación y crecimiento académico.

Edy Francisco Jiménez Merino

Índice de contenidos

Índice de tablas:

Índice de figuras:

Índice de anexos:

1. Título

Modelo QA basado en DistilBERT para responder a preguntas sobre el contenido extraído de tareas académicas de la carrera de Computación de la UNL.

QA model based on DistilBERT to answer questions about content extracted from academic assignments of the Computer Science course at UNL.

2. Resumen

La adaptación de modelos pre-entrenados Question Answering (QA) es una tarea esencial para que estos puedan ser implementados en diferentes escenarios. El objetivo de esta investigación es obtener el valor de la métrica ROUGE al aplicar la técnica Fine-Tuning sobre el modelo DistilBERT para dar respuesta a preguntas sobre el contenido extraído de tareas académicas de la Carrera de Computación de la Universidad Nacional de Loja. Para desarrollar este trabajo se usó la metodología CRISP-ML(Q) como marco de referencia, haciendo uso de sus cuatro primeras fases, en las que se realizó: una recopilación de 30 tareas académicas obtenidas de 6 materias diferentes, de las que se generó 80 preguntas sobre su contenido a través de crowdsourcing, las cuales sirvieron como base para crear un dataset en formato SQuAD1.0 con 1410 datos, de los cuales 800 se generaron mediante paráfrasis y el enfoque Few-shot learning, y los 610 restantes con el aporte directo del autor, este dataset se dividió en 90% para entrenamiento (train) y 10% para evaluación (test), con una subdivisión adicional del conjunto train (75% train y 25% validation), teniendo los datos preparados se ajustó hiperparámetros de DistilBERT para entrenar cuatro modelos diferentes usando TensorFlow en la plataforma Google Colab con el entorno de ejecución GPU T4, seleccionando el mejor modelo en base a su nivel de extracción de respuestas y F1-score. Una vez elegido el modelo QA, se realizó una evaluación mediante la métrica ROUGE incluida una prueba A/B testing. El modelo QA se desplegó en Hugging Face y logró una precisión de 86,93% durante su entrenamiento con 51 épocas, learning_rate de $1e^{-5}$ y batch_size de 32, el cual mediante la evaluación logró un F-measure máximo en ROUGE-L de 60,96. Estos valores demuestran la importancia de aplicar el Fine-Tuning en el desarrollo de modelos QA para contextos específicos.

Palabras Clave: *modelo QA, DistilBERT, dataset SQuAD1.0, CRISP-ML(Q), ROUGE*

Abstract

The adaptation of pre-trained question-answering (QA) models is an essential task so that they can be implemented in different scenarios. The objective of this research is to obtain the value of the rough metric by applying the Fine-Tuning technique to the DistilBERT model to answer questions about the content extracted from academic tasks of the Computer Science Department of the National University of Loja. To develop this work, the CRISP-ML (Q) methodology was used as a reference framework, making use of its first four phases, in which the following was done: a compilation of 30 academic tasks obtained from 6 different subjects, from which 80 questions about their content were generated through crowdsourcing, which served as a basis for creating a dataset in SQuAD1.0 format with 1410 data, of which 800 were generated through paraphrasing and the Few-shot learning approach, and the remaining 610 with the direct contribution of the author. This dataset was divided into 90% for training (train) and 10% for evaluation (test), with an additional subdivision of the train set (75% for train and 25% for validation). Having the data prepared, DistilBERT hyperparameters were adjusted to train four different models using TensorFlow on the Google Colab platform with the GPU T4 runtime environment, selecting the best model based on its level of response extraction and F1-score. Once the QA model was chosen, an evaluation was performed using the ROUGE metric, including A/B testing. The QA model was deployed in Hugging Face and achieved an accuracy of 86.93% during its training with 51 epochs, a learning rate of $1e$ -5, and a batch size of 32, which through evaluation achieved a maximum F-measure in ROUGE-L of 60.96. These values demonstrate the importance of applying Fine-Tuning in the development of QA models for specific contexts.

Keywords: QA model, DistilBERT, SQuAD 1.0 dataset, CRISP-ML(Q), ROUGE

3. Introducción

Los modelos multimodales como BERT[1], GPT[2], entre otros, han revolucionado el procesamiento de lenguaje natural (PLN). Estos modelos realizan múltiples tareas como: responder preguntas, procesar imágenes, generar texto, analizar datos, realizar predicciones y muchas actividades más de manera sobresaliente. Sin embargo, a pesar de estos avances en PLN, la mayoría de estos modelos están diseñados y enfocados en la generalidad lo que afecta su eficiencia en contextos particulares. En el ámbito académico, esta falta de especialización limita su efectividad especialmente en los sistemas de búsqueda de respuestas o más conocidos como Question Answering (QA).

Actualmente, no existen conjuntos de datos (datasets) específicos en idioma español para adaptar modelos QA a escenarios o contextos específicos, como es el caso de la carrera de Computación de la Universidad Nacional de Loja (UNL). Estos modelos realizan su entrenamiento y la respectiva evaluación utilizando datasets estándar como SQuAD en su primera o segunda versión, conjunto de datos que no reflejan los contenidos específicos[3], [4] de las tareas o documentos académicos de la carrera antes mencionada. Esta brecha de información y especialización en las investigaciones representa un desafío significativo, ya que las métricas de evaluación de modelos QA como ROUGE (Recall-Oriented Understudy for Gisting Evaluation) pueden verse comprometidas al aplicar estos modelos en contextos especializados sin la debida adaptación[5]. El valor significativo de abordar este problema radica en la necesidad de contar con un modelo QA ajustado y entrenado con su propio dataset, el cual tiene la capacidad responder a preguntas sobre el contenido específico, mejorando así la pertinencia de las respuestas generadas sobre información de tareas académicas de la carrera de Computación de la UNL. Además, el desarrollo de este modelo ofrece beneficios directos al sector educativo, facilitando el acceso a información para la comunidad académica. Este estudio a diferencia de investigaciones previas[6], [7] en el ámbito del procesamiento de lenguaje natural se enfoca en un contexto académico específico en idioma español. El desarrollo de este trabajo se basa en el uso de la metodología CRISP-ML(Q) como un framework (marco de trabajo o marco de referencia) haciendo uso de sus cuatro primeras fases: comprensión del negocio y los datos, ingeniería de datos (preparación de datos), ingeniería del modelo y evaluación del modelo[8].

El presente Trabajo de integración curricular (TIC) tiene como finalidad responder a la pregunta de investigación: **¿Qué puntuación ROUGE se obtiene al ajustar un modelo QA implementando la técnica Fine-Tuning en DistilBERT para dar respuesta a preguntas sobre el contenido extraído de tareas académicas de la carrera de Computación en la UNL?**, por lo que para dar respuesta a la pregunta mencionada se traza el objetivo general:

“**Determinar la puntuación ROUGE que se obtiene al ajustar un modelo QA implementando la técnica Fine-Tuning en DistilBERT para dar respuesta a preguntas sobre el contenido extraído de tareas académicas de la carrera de Computación en la UNL**”, complementado por dos objetivos específicos: “**Adaptar el modelo DistilBERT para responder a preguntas sobre el contenido extraído de tareas académicas en la carrera de Computación de la UNL**” y “**Evaluar el rendimiento del modelo mediante el análisis de la métrica ROUGE en la obtención de respuestas dadas por el modelo QA**”.

El alcance de este TIC se centra en la obtención y evaluación de un modelo de QA basado en el ajuste de DistilBERT, con el propósito de responder a preguntas sobre el contenido extraído de tareas académicas de la carrera de Computación en la Universidad Nacional de Loja, utilizando la métrica ROUGE para evaluar su desempeño. Una de las principales limitaciones fue la obtención de datos académicos específicos (documentos, tareas, indicaciones, entre otros) y la creación del dataset, el cual requirió una considerable inversión de tiempo. Mismo es el caso de la generación de preguntas mediante crowdsourcing y la creación de datos que complementaron el dataset para entrenar el modelo QA. Además, el Fine-Tuning (ajuste fino) de hiperparámetros de DistilBERT y las múltiples iteraciones y pruebas incrementaron el uso de recursos computacionales, presentando limitaciones con el entorno de ejecución GPU T4 de Google Colab.

4. Marco Teórico

4.1. Antecedentes

En la actualidad, la aplicación de modelos de procesamiento del lenguaje natural (PLN) multimodales, como BERT (Bidirectional Encoder Representations from Transformers) [1], GPT (Generative Pre-trained Transformer)[2], entre otros, ha transformado por completo la comprensión del lenguaje por parte de las máquinas, en especial en modelos de búsqueda de respuestas en inglés question answering (QA). Estos modelos pre-entrenados han demostrado su eficacia en diferentes escenarios y aplicaciones tales como: asistentes virtuales, sistemas de búsqueda, atención al cliente, motores de búsqueda, etc. Investigaciones previas, como el proyecto "ChatBot para preguntas sobre el COVID-19"[7], han explorado el potencial de modelos de lenguaje en este caso BERT para responder a preguntas en escenarios específicos como lo es el covid-19. Asimismo, se han desarrollado chatbots basados en DistilBERT una versión ligera de BERT, como se describe en el artículo "Chatbot basado en una versión ligera del modelo BERT para resolver inquietudes relacionadas con matrículas y homologaciones en la Universidad Nacional de Loja"[6], demostrando la versatilidad y aplicabilidad de un enfoque de ajuste para adaptar modelos a entornos específicos y sobre todo académicos[3].

En Ecuador, específicamente en la Universidad Nacional de Loja (UNL), se mantiene un proyecto de vinculación con la sociedad que tiene como nombre "Inclusión lectora de estudiantes con discapacidad visual mediante innovación tecnológica" (vea **Anexo 1**), del que surge la necesidad de implementar un módulo de búsqueda de respuestas sobre documentos como: tareas, indicaciones, actividades, etc, tal y como lo evidencia su director el Ing. Oscar Miguel Cumbicus Pineda (ver pregunta 1 sección Componentes/módulos del **Anexo *2***), el cuál menciona que: se requiere un modelo de QA capaz de responder a consultas sobre el contenido extraído de estos documentos académicos. Este módulo es un componente clave para el éxito del proyecto antes mencionado en la UNL. Por consiguiente, el uso de modelos pre-entrenados con arquitecturas Transformers, como DistilBERT alimentado con su propio dataset[1], se presenta como una tarea efectiva para abordar esta necesidad en este contexto específico. Integrar DistilBERT en el proyecto de vinculación no lo solo permite responder a preguntas sobre el contenido extraído de documentos, sino que también presenta una alternativa a procesos manuales como revisar todo un corpus de texto para encontrar la información que se necesita.

[1] Dataset: conjunto de datos

4.2. Fundamentación Teórica

4.2.1. Inteligencia Artificial

La inteligencia artificial (IA) o AI por sus siglas en inglés, es un conjunto de tecnologías que permiten el uso de opciones avanzadas dentro de un equipo, sistema o herramienta informática, como la capacidad de describir imágenes, traducir automáticamente, responder a preguntas sobre un determinado contexto, analizar datos, entre otras [9]. La IA en la actualidad es uno de los instrumentos con mayor apogeo para la innovación en múltiples disciplinas como: informática, estadística, ingeniería, lingüística, neurociencia, filosofía y psicología, siendo una herramienta altamente demandada [10] por empresas, instituciones, emprendimientos, etc. Por ejemplo, la mayoría de chatbots funcionan automáticamente mediante el uso de modelos de búsqueda de respuestas más conocidos como modelos Question Answering (QA) para brindar respuesta a inquietudes frecuentes sin la necesidad de intervención humana, al no ser en su creación y mantenimiento.

La IA[2] es una de las tecnologías emergentes cuyo enfoque está en la creación de un sistema con capacidad similar a la del ser humano en una o más habilidades, estas pueden ser: razonar, aprender, actuar o responder de una forma que normalmente necesitaría inteligencia humana, con ello realizar actividades de forma automática reduciendo el tiempo y generando resultados de forma masiva en diversos campos[11]. La IA se está convirtiendo en una de las tecnologías más disruptivas[3] de esta época contemporánea debido al uso y dependencia que ha generado en muchas de las actividades cotidianas que realizan las personas en cualquier parte del mundo. Esto se debe a su capacidad de automatización y su facilidad de uso, lo que se traduce en el ahorro y optimización de tiempo en realizar actividades específicas, además la IA ofrece múltiples ventajas. En la **Tabla *1***, se puede evidenciar las ventajas más relevantes de la IA en la actualidad.

Tabla 1. *Ventajas de la Inteligencia artificial en la actualidad*

Ventajas de la Inteligencia Artificial		
Reduce el error humano	**Elimina las tareas repetitivas**	**Automatización**
Es capaz de eliminar los errores manuales en: el procesamiento o análisis de datos, el montaje, en la fabricación, entre otras.	Puede usarse para llevar a cabo tareas repetitivas, lo que brinda el espacio y tiempo a problemas de mayor impacto.	Es capaz de automatizar flujos de trabajo y procesos, o bien operar de manera autónoma e independiente.
Rápida y precisa	**Disponibilidad infinita**	**Investigación y desarrollo agilizados**
Puede procesar información más rápido y realizar búsqueda de patrones de forma precisa	Las herramientas no tienen obligaciones, por lo que su disponibilidad es “infinita”	Alta capacidad de análisis de datos, esto ayuda a optimizar proyectos e investigaciones

Nota: Esta es una visión general de las posibles alternativas y soluciones que ofrece la IA

[2] Google Cloud: ¿Qué es la inteligencia artificial o IA?
[3] Tecnologías disruptivas: son innovaciones que transforman significativamente de forma positiva o negativa un sector existente

4.2.2. Machine Learning

El machine learning en español aprendizaje automático, conocido también como "ML" por sus siglas en inglés, es un término que surgió en la década de los sesenta (1960)[12], es una rama de la Inteligencia artificial que se enfoca en aspectos como: desarrollo, construcción o ajuste de algoritmos y modelos de IA, entre otras funciones, mismos que se implementan en sistemas o herramientas computacionales para llevar a cabo tareas automáticamente, basándose en patrones e inferencias realizadas a través de un entrenamiento y validación previas. Estos sistemas por lo general durante su entrenamiento procesan grandes cantidades de datos para identificar patrones estadísticos y generar resultados lo más preciso posible a partir de datos de entrada[13]. Por ejemplo, en la conducción autónoma, los sistemas de ML se utilizan para analizar en tiempo real las imágenes y detectar otros vehículos en la carretera, con ellos predecir su movimiento y así evitar accidentes. El ML está formado de diversas técnicas y métodos, de los cuales los más imprescindibles son:

- **El aprendizaje supervisado**: Este método se enfoca en el uso e implementación algoritmos que aprenden a partir de un entrenamiento previo con un conjunto de datos etiquetados para realizar tareas como: clasificación y regresión.
- **El aprendizaje no supervisado:** Este método se aplica mediante algoritmos para identificar patrones y relaciones en conjuntos de datos que a diferencia del aprendizaje supervisado estos se encuentran sin etiquetar, es utilizado comúnmente en clustering, reducción de dimensionalidad, entre otros.
- **El aprendizaje por refuerzo:** Esta técnica se basa en la toma de decisiones mediante sistemas de recompensa-castigo, esto intenta maximizar una métrica específica de rendimiento del sistema de ML. Dentro de esta categoría se encuentra el deep learning (aprendizaje profundo), un enfoque basa su aprendizaje al uso de redes neuronales artificiales con múltiples capas, demostrando así un alto rendimiento para aprender representaciones jerárquicas de conjuntos de datos de imágenes, texto y audio[14], impulsando avances significativos y sobresalientes en áreas como: visión por computadora, procesamiento de lenguaje natural (PLN) e inteligencia artificial en general.

Durante las últimas décadas, los avances tecnológicos dispositivos de procesamiento han permitido el desarrollo de sistemas más avanzados y eficientes basados en ML, además el ML se usa para entrenar algoritmos de clasificaciones, predicciones y descubrimientos clave en proyectos de minería de datos[15].

4.2.3. Deep Learning

El Deep Learning más conocido como aprendizaje profundo, es un campo avanzado derivado del Machine Learning que ha revolucionado el procesamiento de lenguaje, sobre todo en la forma en que las máquinas procesan y analizan grandes cantidades de datos para una tarea específica[16], [17]. Esta técnica se enfoca en la creación de redes neuronales artificiales con varias capas, capaz que su funcionamiento se aproxime a la estructura, destreza y funcionamiento del cerebro humano[17], permitiendo así que una máquina aprenda de manera autónoma y sea capaz de tomar decisiones sin necesidad de ser programada cada vez que se requiera realizar una tarea o actividad, solamente para el mantenimiento y agregación de nuevas funcionalidades o mejoras.

El Deep Learning basa su funcionamiento en el entrenamiento mediante algoritmos de optimización, backpropagation[4] y técnicas de regularización para evitar el sobreajuste (overfitting), estos procesos requieren grandes cantidades de datos de entrenamiento y sobre todo de recursos computacionales (GPUs u otros), especialmente para tareas complejas y que requieren gran capacidad de procesamiento[18]. Sin embargo, el Deep Learning ha demostrado ser fundamental en la resolución de problemas complejos especialmente en la creación de sistemas inteligentes capaces de adaptarse y aprender de manera automática.

Las aplicaciones del Deep Learning en la actualidad son diversas y son multidisciplinarias, algunos ejemplos son: el reconocimiento de patrones (reconocimiento de objetos, detección facial, reconocimiento de voz, etc), el procesamiento del lenguaje natural (traducción automática, análisis de sentimientos, generación de texto y los modelos de búsqueda de respuestas), visión por computadora (detección de objetos, segmentación de imágenes, reconstrucción 3D), entre otros[19].

4.2.4. Procesamiento de Lenguaje Natural

El procesamiento de lenguaje natural (PLN) o en inglés Natural Language Processing (NLP), es un subcampo de la IA que se enfoca en el desarrollo de modelos y algoritmos capaces de: analizar, comprender, extraer, procesar y generar lenguaje humano, mediante un entrenamiento previo[20]. Su objetivo principal es permitir una interacción entre las personas y los ordenadores, esto mediante una comunicación realizada en lenguaje natural. Una de las tareas más sobresalientes del PLN es el análisis semántico, mismo que implica la extracción del significado y la comprensión de un contexto. Esto involucra técnicas como: análisis sintáctico, eliminación de ambigüedades en las palabras y el reconocimiento de entidades. Otra de las tareas principales del PLN es la generación y extracción de texto, que se encarga de producir lenguaje natural de forma comprensible y extraer información de forma clara.

[4] Backpropagation: es un algoritmo que se usa para entrenar redes neuronales artificiales para minimizar los errores en el proceso de aprendizaje automático

Los modelos de PLN se basan en técnicas de aprendizaje automático y aprendizaje profundo, tales como: redes neuronales Transformers, los modelos probabilísticos y CNN[21]. Estos modelos son entrenados con grandes cantidades de datos de texto, como: libros, artículos, transcripciones, incluso la web como el corpus Wikipedia, esto para aprender las reglas gramaticales, el vocabulario y el contexto del lenguaje humano, muchas de las fuentes de datos para el entrenamiento estos modelos son los datasets estructurados de forma que puedan ser utilizados en el algoritmo que se haya seleccionado para determinada tarea. El procesamiento de lenguaje natural tiene numerosas aplicaciones en diversos campos, por ejemplo: traducción automática, los asistentes virtuales, modelos de búsqueda de respuesta y resúmenes automáticos, incluso en los sistemas de reconocimiento de voz y el análisis de sentimientos[22]. A medida que los modelos de PLN se vuelven más sofisticados, la interacción entre el ser humano y la máquina es directa.

4.2.5. Sistemas Question Answering (QA)

Los sistemas pregunta-respuesta o sistemas de búsqueda de respuestas más conocidos como “Question Answering”(en inglés), son un tipo de sistemas automáticos que se usan para recuperar o extraer información de un fragmento de texto, esto mediante el entrenamiento utilizando un conjunto de datos estructurado llamado comúnmente como dataset, o haciendo uso de una colección de documentos e información (como World Wide Web), buscan brindar una respuesta clara y precisa acerca de una pregunta generada en lenguaje natural por un usuario[30].En la **Figura 1** se presenta el flujo de información de un Sistema QA, el cual sirve para generar una respuesta a partir de una pregunta y su respectivo contexto.

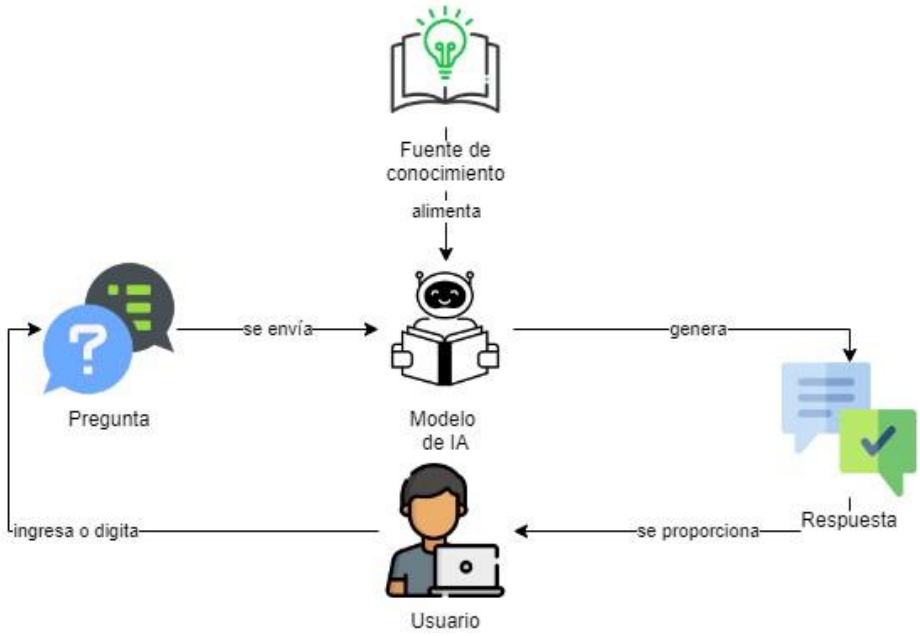


Figura 1. *Flujo de información e interacción entre los componentes de un sistema QA.*

4.2.6. Modelos de Lenguaje de Gran Escala

Los Modelos de Lenguaje de Gran Escala (Large Language Models o LLMs, por sus siglas en inglés), son programas informáticos al que se le han dado suficientes ejemplos para que sea capaz de reconocer e interpretar el lenguaje del ser humano u otros tipos de datos complejos, utilizan el aprendizaje profundo para entender cómo funcionan los caracteres, las palabras y las frases en conjunto. Estos modelos se utilizan para analizar y procesar texto, imágenes, documentos, entre otros de manera automatizada. Un hito clave de estos modelos LLMs fue la introducción del modelo BERT[23] y GPT[5]. Ambos modelos han sido entrenados en grandes corpus de datos e información de texto, lo que les ha permitido procesar de forma general una comprensión profunda del lenguaje de las personas.

4.2.6.1. BERT.

BERT (Bidirectional Encoder Representations from Transformers) o traducido al español “Representación de Codificador Bidireccional de Transformers” es un tipo de modelo LLMs desarrollado por Google que ha transformado el campo del PLN. Su arquitectura es de tipo Transformer, un tipo de red neuronal diseñada para procesar secuencias de datos de manera eficiente. BERT utiliza una arquitectura codificador-decodificador y es bidireccional[24], está diseñado para capturar representaciones bidireccionales profundas a partir de texto sin etiquetar condicionando conjuntamente el texto izquierdo y derecho en todas las capas. Como resultado, el modelo BERT pre-entrenado puede ajustarse con sólo una capa de salida adicional y así crear, desarrollar o ajustar modelos derivados para una amplia gama de tareas, como la respuesta a preguntas y la inferencia lingüística, sin modificaciones sustanciales de la arquitectura específica de la tarea. Sin embargo, entrenarlo requiere de muchos recursos, por lo que actualmente se ha procedido a crear modelos derivados, como es el caso de DistilBERT, RoBERTa, etc.

4.2.6.2. DistilBERT.

DistilBERT es una versión más ligera de BERT, fue desarrollada por Victor Sanh, Lysandre Debut y Thomas Wolf en Hugging Face[25], utiliza técnicas de destilación de conocimientos cuyo objetivo es reducir el tamaño y la complejidad de la arquitectura original de un modelo más grande, manteniendo un rendimiento comparable en muchas tareas de PLN. Esta arquitectura conserva hasta en un 97% de las prestaciones de BERT, mientras que su tamaño se reduce hasta llegar a un 40% de tamaño y logra ser hasta un 60% más rápido[26]. Estas características hacen que DistilBERT sea una alternativa idónea en

[5] ChatGPT

escenarios específicos donde se requiera un equilibrio entre la precisión y la eficiencia computacional, demostrando ser especialmente útil en aplicaciones o contextos donde sus recursos son limitados.

DistilBERT se destaca por su capacidad para pre-entrenar un modelo más pequeño, que de acuerdo a diversos factores puede ser ajustado y enfocado con buen rendimiento en una amplia gama de tareas, manteniendo un tamaño reducido y una latencia inferior[27]. Este proceso de aproximación, conocido como destilación, ha permitido operar modelos grandes escenarios con pocos recursos, lo que representa un aporte sobresaliente; demostrando ser una herramienta valiosa al mantener un alto rendimiento en tareas de comprensión del lenguaje, al tiempo que reduce significativamente la complejidad y el tamaño del modelo, mantiene una estructura similar a la BERT[28]. En la **Figura 2** se presenta la arquitectura y componentes principales del modelo DistilBERT[29], donde la primera capa representa la entrada que dispone de mecanismos de atención, mismos que se realizan en conjunto con la normalización; la segunda capa consta de los componentes de BERT que, a través del ajuste, en este caso la “distilation” y del tokenizador de texto, se obtiene un modelo comprimido llamado DistilBERT componente que forma la última capa, al igual que BERT, también devuelve una salida(respuesta).

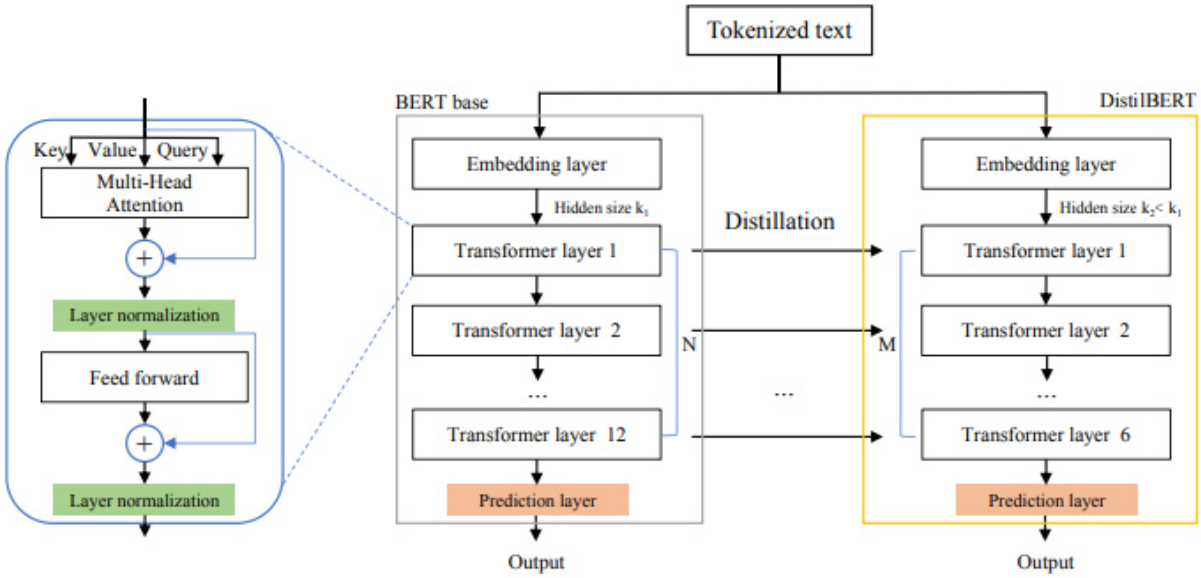


Figura 2. *Arquitectura y componentes generales de DistilBERT. Imagen obtenida de* [22]

- **Tokenizador**

Es una herramienta de los modelos de PLN que se utiliza para dividir un texto en unidades discretas llamadas "tokens" que representan la esencia de la información origina, estos tokens pueden ser símbolos, palabras y otra representación que conserve algo del formato inicial, esto depende del enfoque utilizado y del idioma del texto, es pocas palabras, se encarga de dividir un texto en unidades o segmentos más pequeños[30]. En el modelo

DistilBERT el tokenizador correspondiente se llama “DistilBertTokenizer[6]”. Este tokenizador tiene la función de preprocesar texto, y adicionalmente incluye la normalización del texto, la eliminación de caracteres especiales y la conversión de palabras a su forma base, permitiendo procesar texto de forma eficiente[31], también maneja la conversión inversa, transformando los tokens procesados nuevamente en texto legible para la interpretación de los resultados, para ello los tokens conservan alguna característica del texto original, para poder regresar a su forma base.

DistilBertTokenizer aplica técnicas como la tokenización subpalabra (WordPiece[7]) que ayuda a procesar las palabras desconocidas(raras) en la información del conjunto de datos de entrenamiento. Esta técnica divide palabras en subcomponentes manejables, permitiendo así que el modelo comprenda y generalice a partir de un vocabulario limitado[32]. Además, este tokenizador añade tokens especiales que indican el inicio y el fin de la secuencia, y gestionando el padding(relleno)[8] para asegurar que todas las secuencias tengan la misma longitud.

- **Hiperparámetros**

Los hiperparámetros son aquellos valores de un modelo con tendencia a ser modificables, caso contrario a los parámetros que no son modificables. Un hiperparámetro es todo aquello que podemos ajustar, “tunear” o modificar para alterar el desempeño de una red, modelo, algoritmo o experimento[33]. A continuación, se describen algunos ejemplos: la arquitectura, el método de regularización, el optimizador, la taza de aprendizaje, las épocas, entre otros son ejemplos de hiperparámetros. En un modelo de ML existen diferentes tipos de hiperparámetros, por lo que a continuación se redactan aquellos que tiene tendencia a producir cambios más impactantes durante el ajuste de DistilBERT.

1. ***Learning Rate (Tasa de Aprendizaje):*** La tasa de aprendizaje es uno de esos hiperparámetros que definen el ajuste en los pesos de un modelo respecto al descenso del gradiente de pérdidas, además, controla la magnitud de las actualizaciones de los pesos del modelo en cada paso, iteración o época. Una tasa de aprendizaje muy alta puede hacer que el modelo no converja (no encuentre su punto óptimo), mientras que una muy baja puede hacer que el entrenamiento sea muy lento en términos de tiempo mas no de rendimiento. Para el learning rate los valores comunes son 1e-5, 2e-5, 3e-5 y 5e-5[34]. Es crucial ajustar este parámetro cuidadosamente, ya que influye significativamente en la velocidad y la estabilidad del entrenamiento.
2. **Epochs (Épocas):** Es una iteración completa a través de todo dataset de entrenamiento(train) en un ciclo para entrenar el modelo de aprendizaje automático.

[6] DistilBertTokenizer
[7] HuggingFace: Tokenización WordPiece
[8] Padding

Durante una época, el modelo procesa todas las muestras disponibles de entrenamiento del conjunto de datos y actualiza sus ponderaciones y sesgos en función de la pérdida o el error calculados, depende directamente de otro hiperparámetro que es el batch_size[35]. Más épocas permiten al modelo aprender mejor, pero también aumentan el riesgo de sobreajuste. Para la cantidad de épocas cantidades comúnmente usadas son 10, 50 y 100 épocas, para establecer la mejor cantidad de este hiperparámetro es importante monitorear el rendimiento en el conjunto de validación para determinar cuándo detener el entrenamiento.

3. **Weight Decay (Decaimiento del Peso):** La regularización weight decay puede considerarse una forma próxima de tratar la regularización, es decir, es tratamiento diferente de la regularización en el método de optimización ayudando a prevenir el sobreajuste[36] y así reduce los pesos ligeramente en cada paso de optimización. En este hiperparámetro los valores comunes son 0.01 y 0.1.
4. **Batch_Size (Tamaño del Lote):** Este hiperparámetro determina cuántos ejemplos de entrenamiento se procesan antes de actualizar los pesos del modelo, se refiere al número de muestras de datos procesadas en conjunto durante el entrenamiento en el aprendizaje automático. En el ML, un tamaño de lote mayor significa que el modelo procesa más datos a la vez, lo que puede acelerar el entrenamiento, pero por lo general requiere más memoria. Un tamaño de lote pequeño puede hacer que el modelo aprenda patrones más ruidosos, mientras que un tamaño de lote grande puede aprovechar mejor el paralelismo y procesamiento del hardware. En este hiperparámetro los valores típicos son: 8, 16 y 32[37]. La elección del tamaño de lote puede verse limitada por la memoria de la GPU disponible o la capacidad de un ordenador en particular.
5. **Seed(semilla):** Las semillas son valores numéricos o vectores aleatorios que se utilizan para inicializar diversos procesos, garantizando que los resultados no sean deterministas y presenten propiedades deseables, se usan para inicializar los generadores de números aleatorios para garantizar la reproducibilidad de los resultados generalmente a la hora de entrenar un modelo de ML. Uno de los valores comunes es 42, aunque cualquier número puede ser utilizado. El establecimiento de la semilla permite que los experimentos sean replicables, produciendo los mismos resultados cada vez que se ejecuta el entrenamiento con el valor de la semilla.

4.2.7. Redes Transformers

Las redes Transformers son un tipo de arquitectura de redes neuronales enfocadas en el PLN, el término “redes transformers”, en inglés “Network transformers”, fue introducido por el científico Ashish Vaswan en una revista publicada por Google en el año 2017 [38], desde ahí su evolución y uso es considerable debido a que estas redes han demostrado un

rendimiento significativo para como la traducción automática, la generación de texto y la búsqueda de respuesta a preguntas (QA). La importancia de las redes Transformer radica en su capacidad, ya que puede capturar relaciones de largo alcance en las secuencias de entrada, superando las limitaciones de las redes neuronales recurrentes y logrando así una precisión considerable[39]. Además, posee un mecanismo de atención lo que le permite un entrenamiento paralelo y eficiente, esto disminuye considerablemente los tiempos de entrenamiento permitiendo un mayor aprovechamiento de los recursos computacionales de hardware y software.

La arquitectura de las redes Transformers consta de dos componentes principales: el codificador (encoder) y el decodificador (decoder). En la **Figura 3** se puede evidenciar los componentes principales de una red neuronal Tranformer en una tarea de PLN.

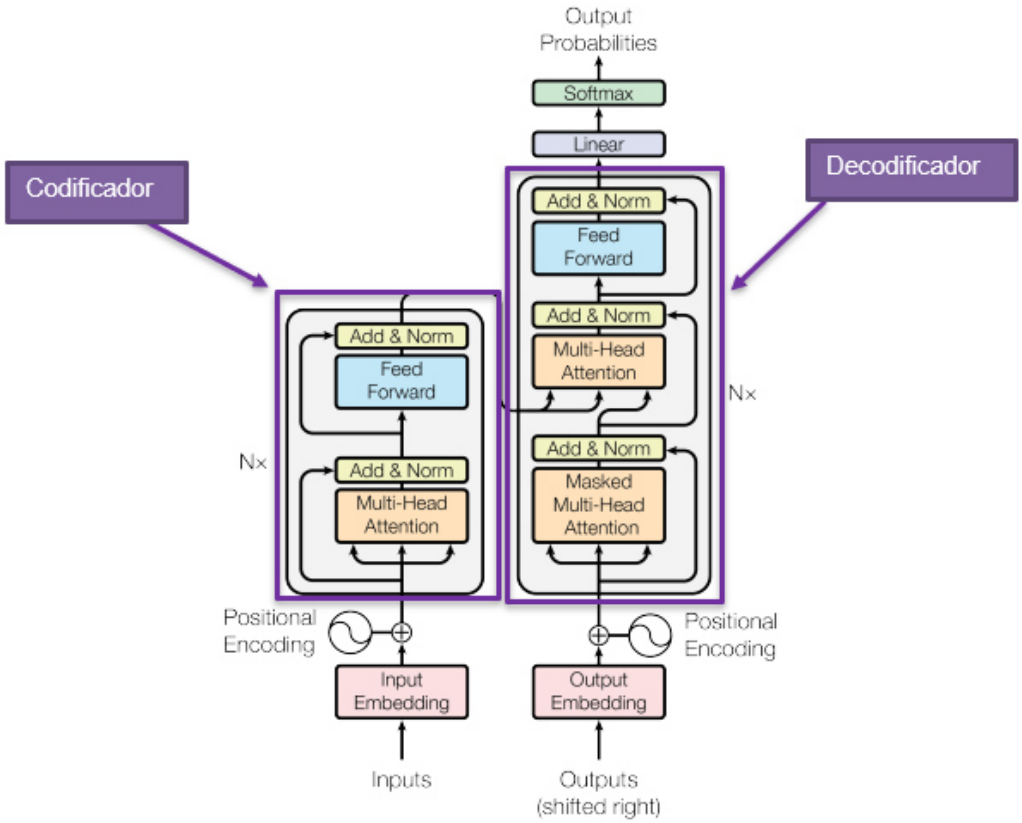


Figura 3. *Arquitectura de una red neuronal Transformer*

Descripción: *Se puede observar dos elementos principales: el codificador (el de la izquierda) y el decodificador (el de la derecha), además se encuentran las entradas, las salidas, los criterios de los mecanismos de atención y las probabilidades. Esta figura está adaptada de* [40]

El encoder se encarga de procesar la entrada y proporcionar una representación interna, el decodificador utiliza esta representación para generar la salida. Como componente central de las redes Transformer tenemos el mecanismo de atención múltiple (Multi-Head Attention[9]), que permite al modelo enfocarse en diferentes partes de la entrada de manera

[9] Multi-Head Attention

simultánea. Tanto el codificador como el decodificador están compuestos por múltiples capas de atención múltiple, donde cada capa captura diferentes relaciones entre los elementos de la entrada. En el codificador, cada una de las capas de atención múltiple procesan la entrada y generan una representación interna llamada "embedding[10]". Esta representación obtiene las relaciones semánticas y contextuales de la entrada. Además, el codificador utiliza un mecanismo de atención de auto-codificación (Self-Attention[11]), el cual permite que cada elemento de la entrada se relacione con los demás elementos, lo que facilita la captura de dependencias de largo alcance[41].

Por otro lado, el decodificador usa dos tipos de atención: la atención de auto decodificación (Self-Attention) y la atención codificador-decodificador (Encoder-Decoder Attention[12]). La primera permite que cada elemento de la salida se relacione con los demás elementos de la salida, mientras que la segunda relaciona la salida con la representación interna generada por el codificador.

4.2.8. Técnicas de los modelos de PLN

El ajuste de modelos PLN requiere de diversos componentes y técnicas, dependen de la actividad o necesidad que se requiera cubrir, a continuación, se mencionan algunas de las técnicas más usadas durante su adaptación en entornos, escenarios y contextos específicos del procesamiento de lenguaje natural.

4.2.8.1. Crowdsourcing.

El crowdsourcing es una técnica que aprovecha la colaboración un grupo de personas, consiste en obtener información u opiniones de un grupo de personas que envían estos datos a través de páginas web colaborativas, redes sociales y aplicaciones para teléfonos inteligentes, otras plataformas que se usan en el contexto del PLN incluyen kaggle, Google docs, entre otras, esto con la finalidad de realizar tareas que tradicionalmente serían llevadas a cabo por un grupo específico de personas (estudiantes, trabajadores, etc)[42]. Este enfoque permite obtener variedad de ideas u opiniones dentro de un campo específico[43].

El crowdsourcing permite de alguna manera acceder a aportes que de otro modo podrían no estar disponible no podrían estar al alcance económico o serían imposibles de obtener. Por lo general implementar esta técnica resulta ser una opción más económica que contratar empleados a tiempo completo, consultores o incluso puede resultar una opción gratuita, ya que muchas contribuciones pueden ser realizadas por voluntarios, esto facilita la adaptación rápida a cambios en la demanda o en la naturaleza de las tareas[44], fomentando

[10] Embedding
[11] Self-Attention
[12] Encoder-Decoder Attention

la innovación al involucrar a una multitud con diferentes perspectivas y conocimientos. El crowdsourcing es multidisciplinario ya que se aplica en diversos campos, un ejemplo claro de ello es la ciencia y la investigación que permiten al público general contribuir con datos y análisis, permitiendo así obtener datasets para tareas como el QA[45], traducciones, generación de resúmenes, etc. En la **Figura 4** se presenta un ejemplo de la técnica crowdsourcing para la elaboración de un dataset QA usando la plataforma Google Docs como medio de colaboración online.

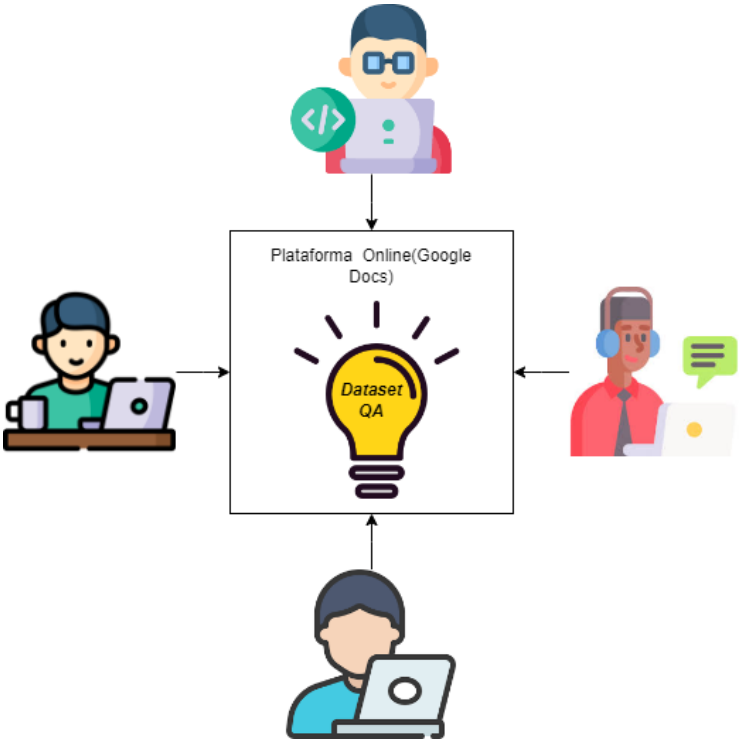


Figura 4. *Representación gráfica de la técnica crowdsourcing en la construcción de un dataset QA*

4.2.8.2. Few-shot Learning.

El Few-shot Learning en español se traduce como “aprendizaje con pocos ejemplos”, es un marco de aprendizaje automático, mediante el cual un modelo de IA logra aprender a hacer predicciones precisas mediante un número reducido de ejemplos etiquetado, suele utilizarse como un método de ajuste en modelos QA o como un enfoque para crear y ampliar un dataset limitado en cuanto a cantidad de datos. Este enfoque contrasta con los métodos tradicionales de aprendizaje automático supervisado, que utilizan corpus de datos para entrenar modelos de IA, en la producción de datos para ampliar un dataset el objetivo principal del Few-shot Learning es abordar el desafío de la escasez de datos en dominios específicos o tareas emergentes[46] por lo que actúa como una técnica de aumento de datos, este a diferencia de las técnicas tradicionales se usa solamente como un enfoque. En muchas situaciones prácticas, recolectar y etiquetar grandes conjuntos de datos puede ser costoso,

consumir mucho tiempo o incluso ser imposible, por ello la capacidad de aprender rápidamente a partir de pocos ejemplos es de suma importancia[47].

Existen varias visiones y métodos para abordar el Few-shot Learning, cada uno con sus propias fortalezas y limitaciones[48], dependen del escenario en el que se encuentre o la necesidad que se necesite abordar, uno de los más populares es el Transfer Learning, el cual aprovecha el conocimiento adquirido por un modelo pre-entrenado en una tarea relacionada y lo transfiere a una nueva tarea en este caso con pocos ejemplos disponibles[49] otro enfoque prometedor es el Fine-Tuning, donde se ajustan una o más capas de un modelo pre-entrenado utilizando los ejemplos de la nueva tarea como puede ser los datos de un dataset[50], con ello el Few-shot Learning busca aprovechar al máximo la información contenida en los pocos ejemplos disponibles para realizar una actualización y ajuste eficiente de los hiperparámetros de un modelo, hay que considerar que el Few-shot Learning enfrenta desafíos importantes, como los que se muestran a continuación:

- Debido a la limitada información se disminuye la capacidad de capturar eficientemente la información relevante[51].
- Los conjuntos de datos pequeños pueden introducir sesgos y variabilidad en los resultados[52].
- No tener suficiente diversidad de datos hace que el modelo o dataset discrimine datos que pueden ser relevantes.

Esta técnica al usarse también como un enfoque de aumento de datos incluye algunos de sus beneficios en los que destaca la generación de datos sintéticos que siguen la estructura y el estilo de los ejemplos originales, es útil cuando se dispone de pocos datos de entrenamiento (train y test) y puede ayudar a diversificar el conjunto de datos.

4.2.8.3. Fine-Tuning.

El Fine-Tuning, más conocido como “ajuste fino”, es una técnica específica dentro del aprendizaje por transferencia o transfer learning que consiste en realizar pequeños ajustes en los hiperparámetros de un modelo pre-entrenado para mejorar su rendimiento en una tarea específica. En lugar de entrenar un modelo desde cero generando toda la complejidad que estos procesos conllevan, además de realizar un alto consumo de recursos computaciones, se toma un modelo pre-entrenado que ha sido entrenado en una gran cantidad de datos y se lo ajusta para que desempeñe una tarea en particular[50], es decir, el Fine-Tuning busca aprovechar el conocimiento y la capacidad de PLN de los modelos grandes como BERT o GPT, incluso más allá con modelos derivados como DistilBERT, RoBERTa, entre otros, y adaptarlos a tareas más específicas. Esto es especialmente útil cuando se dispone de un

conjunto de datos limitado para entrenar y evitar realizar un proceso complejo que conlleva crear un modelo desde cero.

El proceso de Fine-Tuning implica tomar el modelo pre-entrenado y adaptar sus hiperparámetros utilizando un conjunto de datos específico (de imágenes, texto, videos, etc) en la tarea deseada, este ajuste conlleva a modificar los pesos y las conexiones internas del modelo para que se adapten mejor a los datos de entrenamiento específicos. En la **Figura 5** podemos ver el proceso de forma general de lo que implica el Fine-Tuning y Transfer Learning sobre un modelo de IA.

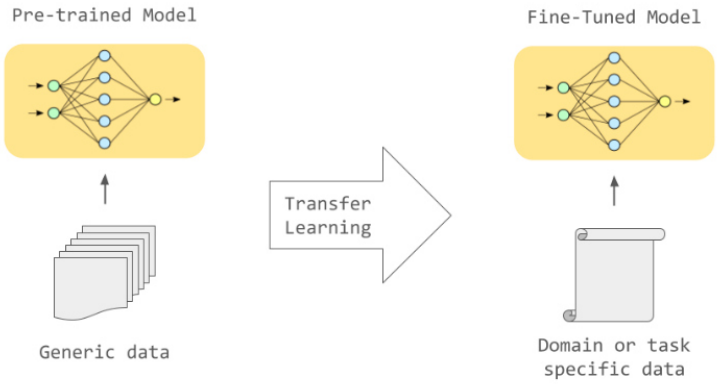


Figura 5. *Proceso general de la técnica Fine-Tuning sobre un modelo pre-entrenado QA.*

Descripción: *En la izquierda tenemos el modelo general entrenado con datos genéricos, el elemento del centro denotado con una flecha representa el proceso Fine-Tuning en el cual mediante un entrenamiento con un dataset específico se obtiene el nuevo modelo que se puede ver a la derecha. Imagen tomada de la página Labellerr*[13]

El Fine-Tuning se realiza en varias etapas, donde primeramente se congela (se mantiene estático) una parte del modelo pre-entrenado, generalmente las capas iniciales que capturan características generales del lenguaje, para posteriormente agregar una o más capas adicionales al modelo para adaptarlo a la tarea específica, finalmente se integran todas las capas para lograr un entrenamiento de todo el conjunto. Durante este entrenamiento, se utilizan técnicas de optimización para ajustar los parámetros del modelo y minimizar la función de pérdida que mide la discrepancia entre las predicciones del modelo y las etiquetas de los datos de entrenamiento. Es importante tener en cuenta la selección de hiperparámetros, como la tasa de aprendizaje, el tamaño del lote, dataset, u otros para obtener un rendimiento significativo con el modelo, con ello a medida que el modelo se entrena, se ajustan los pesos y las conexiones para mejorar su capacidad[53].

[13] Página labellerr

4.2.8.4. Transfer Learning.

El aprendizaje por transferencia, o “transfer learning”, es un enfoque del aprendizaje automático en el que un modelo con un entrenamiento previo en una tarea se utiliza como punto de partida para un modelo en una nueva tarea, transfiriendo el conocimiento que el primer modelo ha aprendido sobre las características de los datos al segundo modelo[49]. Este enfoque es particularmente útil en el campo del PLN donde los modelos de LLMs, por ejemplo, como BERT y su versión destilada DistilBERT, pueden ser pre-entrenados en enormes corpus de texto y luego afinados para tareas específicas. En el contexto de un modelo de QA, el transfer learning permite al modelo pre-entrenado aprovechar el conocimiento general del lenguaje y la comprensión semántica que ha adquirido durante su entrenamiento previo, siendo esto esencial para la capacidad del modelo de comprender las preguntas y las respuestas[54], teniendo en cuenta que esto depende directamente del dataset usado durante el entrenamiento, lo que ayuda a reducir drásticamente el tiempo y los recursos necesarios para entrenar un modelo efectivo, ya que una parte significativa del aprendizaje ya ha sido realizada. En lugar de aprender desde cero cómo procesar y entender el lenguaje, el modelo toma como base el conocimiento que ya ha adquirido, a menudo esto resulta en un mejor rendimiento[55].

4.2.9. Conjunto de datos SQuAD.

Un dataset es un conjunto de datos, que pueden ser estructurados o no estructurados de forma tal que se pueda utilizar en cierto campo y que reflejen la información de dicho contexto, SQuAD, abreviatura de “Stanford Question Answering Dataset”, es un dataset ampliamente utilizado en la investigación de modelos QA[56], comúnmente se encuentra en formato “.json” para mejorar su procesamiento, aunque, en diferentes plataformas como Kaggle o HugginFace se lo ve representado en tablas para disminuir su complejidad ante el lector. Este conjunto de datos se forma de cinco atributos(variables): id (dato alfanumérico), title (dato String), context (dato String), question (dato String) y answers(diccionario), originalmente SQuAD tiene 2 versiones y ambas en idioma inglés, aunque en la actualidad se puede encontrar en idioma español[57]. Este dataset se creó con datos de Wikipedia[14].

Cabe destacar que, para el desarrollo de un dataset en contextos específicos, la fuente de información puede cambiar a voluntad y necesidad de los usuarios[58]. El objetivo es proporcionar un conjunto de datos realista para evaluar la capacidad de los modelos de QA al comprender y responder preguntas basadas en un contexto dado y se divide en 2 versiones que son: SQuAD 1 y SQuAD 2.

[14] Wikipedia: una enciclopedia de información

- **SQuAD 1.0:** En esta versión cada pregunta tiene una respuesta específica que se puede encontrar directamente en el contexto proporcionado en la variable “context”, es decir, la respuesta a la pregunta es extraída del contexto o información proporcionada. El principal desafío para los modelos de QA en SQuAD 1.0 es la extracción precisa de la respuesta, por lo que se debe tener un dataset de calidad y diverso, no necesariamente en cantidad si no en la variabilidad de datos. Aquí, cada entrada contiene valores con el tipo de dato correspondiente en cada una de las columnas (variables o atributos).
    - **SQuAD 2.0:** Es una evolución de SQuAD 1, donde además de preguntas con respuestas que deben encontrarse en la variable “context”, incluye preguntas para las cuales no hay una respuesta explícita en el texto. Esta versión presenta un desafío al requerir que los modelos QA manejen preguntas sin respuesta, evaluando así su capacidad para reconocer la falta de información ante una pregunta dada por un usuario.

El formato de los datos en SQuAD al ser JSON, ayuda al intercambio de datos ligero y fácil de leer, aunque también se presenta en formato “,csv”. En la **Tabla *2*** se muestra un extracto con valores en cada uno de los atributos que componen un dataset en formato SQuAD1.0.

Tabla 2. *Ejemplo de la estructura del dataset SQuAD1.0*

SQuAD1.0				
id	**title**	**context**	**question**	**answers**
5727dd864	TIC	La autoría implica que cualquier problema legal que surja del contenido del informe, incluyendo el plagio, sería responsabilidad del autor (el estudiante), no de la universidad (UNL) o sus representantes.	¿Cuál es el individuo responsable si surge alguna cuestión legal derivada del contenido del informe, como el plagio?	{‘answer_start’: [107], ‘text’: [‘sería responsabilidad del autor (el estudiante)’]}

Nota: el id representa el identificador, el title muestra el tema general, el context detalla el texto del que se va a realizar la pregunta, la question es la duda o incógnita que se quiere responder y la variable answers representa el texto de respuesta y el índice de inicio de la misma.

4.2.10. Conceptos Importantes

Para la elaboración del TIC (Trabajo de Integración Curricular) se abordó diferentes conceptos y herramientas adicionales, mismos que se exponen a continuación.

4.2.10.1. Data Augmentation (Aumento de datos)

Es una técnica utilizada en el ML y PLN para incrementar el tamaño y la diversidad de un conjunto de datos[59]. Esto se logra mediante la creación o generación de nuevas muestras a partir de una o más existentes, aplicando diversas transformaciones que preservan el

significado y la estructura del texto original, estas muestras pueden ser imágenes, texto (preguntas, oraciones, frases), videos, entre otras. El propósito principal del aumento de datos es mejorar la capacidad de un modelo para generalizar y manejar variaciones en el lenguaje, reduciendo el riesgo de sobreajuste[60], siendo especialmente útil en casos donde se dispone de un conjunto de datos limitado o complejo para obtener los datos necesarios, estas cantidades limitadas de datos en muchos de los casos no es suficiente para entrenar eficazmente un modelo de IA, por lo que resulta necesario utilizar una técnica de aumento de datos, misma que se utiliza en tareas como la clasificación de texto, el reconocimiento de entidades, la traducción automática y la generación de texto, entre otras. El aumento de datos en la generación de texto se la realiza mediante la inclusión de sinónimos, omisión de palabras, uso de modelos de ML incluyendo herramientas de inteligencia artificial generativa (IAG), enfoques como el Few-shot Learning, y la paráfrasis que incluye varios de los aspectos mencionados anteriormente.

La paráfrasis es una técnica que implica reescribir palabras, oraciones, frases o fragmentos de texto utilizando estructuras diferentes, con la consigna de conservar el significado original, creando así variaciones del mismo contenido, siendo útil para ampliar datasets. [61].

4.2.10.2. Optimizador Adam.

El optimizador Adam abreviatura de "Adaptive Moment Estimation", que en español se traduce como “Estimación Adaptativa de Momentos”, es un algoritmo de optimización iterativo para minimizar la función de pérdida durante el entrenamiento de redes neuronales, utilizado en el entrenamiento de modelos de ML como los modelos de Question Answering (QA). Este optimizador combina las ventajas de dos métodos populares: el método de Momentum y el algoritmo RMSProp[62], su función principal radica en ajustar iterativamente los hiperparámetros de un modelo minimizando la pérdida en muchos de los casos, mejorando así el rendimiento del modelo en la tarea de aprendizaje específica[63]. Adam calcula estimaciones adaptativas de momentos de primer y segundo orden. En la **Figura 6** se visualiza las fórmulas[15] en las que basa su funcionamiento el optimizador Adam.

[15] Fórmula del Optimizador Adam en Deep learning

$$m = \beta_1 * m + (1 - \beta_1) * \Delta W$$
$$v = \beta_2 * v + (1 - \beta_2) * \Delta W^2$$
$$W = W - \frac{\alpha * m}{\sqrt{v + \varepsilon}}$$

$Donde$:

m: estimación del primer momento (la media móvil de los gradientes)

β_1: Coeficiente de decaimiento exponencial para el primer momento (cerca de 0.9).

ΔW: Gradiente de la función de pérdida con respecto a los parámetros $\boldsymbol{W}$

v: estimación del segundo momento (la media móvil de los cuadrados de los gradientes).

β_2: Coeficiente de decaimiento exponencial para el segundo momento (cerca de 0.999).

W: Parámetros del modelo que se están actualizando

α: Tasa de aprendizaje

ε: Término de estabilidad numérica que evita la división por cero (valor pequeño 10^{-8})

Figura 6. *Fórmulas del Optimizador Adam*

Sin embargo, estas fórmulas al ser de alta complejidad muchas de las veces no se ejecutan de forma manual debido a la facilidad de usar herramientas que realizan es actividad, de hecho, en el ajuste de un modelo QA se utiliza directamente la función *create_optimizer* de la biblioteca *transformers,* la cual viene configurada por defecto en entornos como Google Colab y de forma local se la puede instalar e importar, ya que esta se implementa en escenarios que demandan gran capacidad de procesamiento.

4.2.10.3. A/B testing.

A/B Testing o también llamado “split testing”, es una técnica comparativa que permite evaluar dos o más variantes de un sistema o componente para determinar cuál ofrece mejores resultados en base a métricas establecidas previamente como parámetros de calificación[64]. Se utiliza ampliamente en marketing digital, diseño de interfaces de usuario, modelos de IA, sistemas web, entre otros, proporcionando una base sólida para decisiones basadas en datos. En el caso de los modelos Question Answering, el A/B Testing es una herramienta que permite comparar el rendimiento de diferentes versiones del modelo en la eficiencia de sus respuestas con las respuestas de referencia, que por lo general se encuentran en el conjunto de datos “test”. Por ejemplo, evaluar dos arquitecturas distintas o variantes del modelo con diferentes configuraciones de hiperparámetros, también se puede evaluar los datos del dataset utilizado durante el entrenamiento, y más eficiente los datos de evaluación.

En el A/B Testing interactúan al menos dos entidades conocidas como sujeto A y B, que pueden ser las versiones de los modelos o las entidades (elementos del A/B Testing), en la que se asegura una evaluación o calificación imparcial del rendimiento del modelo. La importancia del A/B Testing en QA radica en su capacidad para validar mejoras,

optimizaciones y la calidad de las salidas generadas[65], lo que permite obtener una apreciación real al medir el impacto de nuevas funcionalidades, técnicas de preprocesamiento, o ajustes en el modelo, asegurando que cualquier cambio introducido efectivamente mejora la precisión y eficiencia del sistema. En la **Figura 7** se puede visualizar una visión holística sobre el método A/B testing, en la que se visualiza dos entidades calificando un modelo QA.

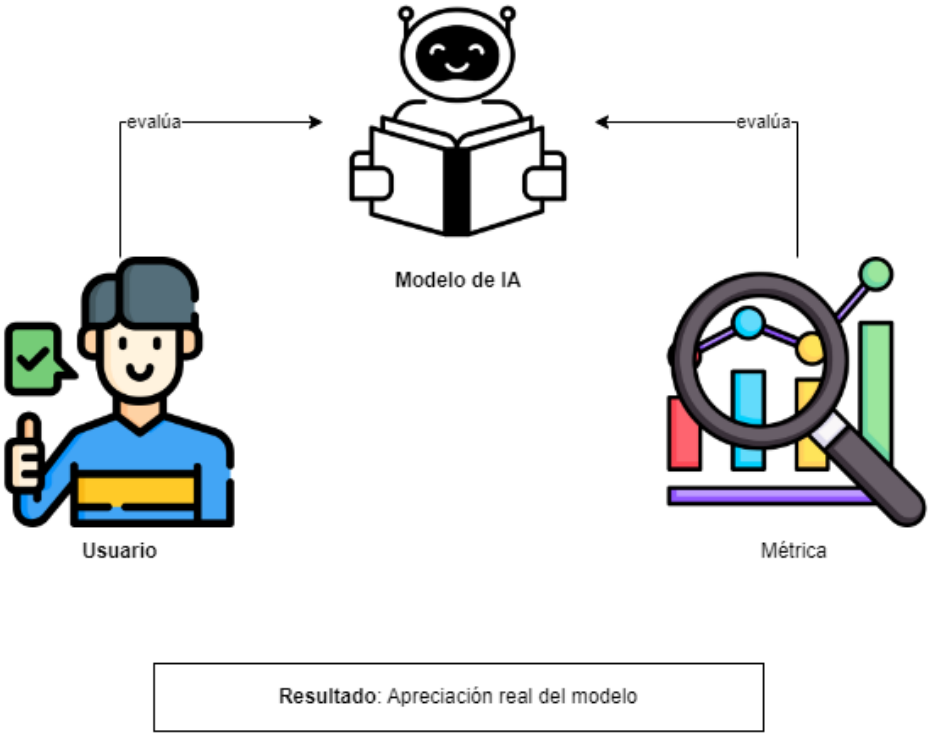


Figura 7. *A/B testing a un modelo de IA*

4.2.10.4. Hugging Face.

Hugging Face fundada en 2016, es una plataforma y comunidad de aprendizaje ML y ciencia de datos que ayuda a los usuarios a construir, desplegar y entrenar modelos de aprendizaje automático, Además, proporciona la infraestructura para la demostración, ejecución y despliegue de modelos de IA, debido a que contiene las funcionalidades necesarias como interfaces para presentar los modelos creados, incluye datasets y herramientas. Los usuarios también pueden navegar por los modelos y conjuntos de datos que otras personas han subido lo que simplifica el proceso de descarga y entrenamiento de

modelos de ML. La biblioteca ofrece a los desarrolladores una forma eficaz de incluir uno de los modelos de ML alojados en Hugging Face en su flujo de trabajo y crear pipelines [16]de ML.

La plataforma es importante por su naturaleza de código abierto y sus herramientas de despliegue, permitiendo el uso de alrededor de 1417 modelos [66] mediante interfaces eficientes para diversas tareas, incluyendo el Question Answering. Además de los modelos, Hugging Face proporciona una plataforma para compartir y acceder a aproximadamente 20 mil datasets.

4.2.10.5. Python.

Python[17] es un lenguaje de programación orientado a objetos, de alto nivel y semántica dinámica, lo que lo hacen atractivo para el desarrollo rápido de aplicaciones, así como para conectar componentes existentes entre sí. La sintaxis de Python, sencilla y fácil de aprender, favorece la legibilidad y, por tanto, reduce el coste de mantenimiento de los programas. Python admite módulos y paquetes, lo que fomenta la modularidad de un programa o sistema y la reutilización del código[67]. Además, cuenta con varias bibliotecas (pandas, numpy, matplotlib, etc) y marcos de trabajo como Django o Flask, es utilizado en aplicaciones de aprendizaje automático, incluyendo modelos QA[68]. Python también se utiliza para preprocesar datos, limpieza de datos, preprocesamiento, normalización y varias acciones más. En la creación o ajuste de modelo QA estas ventajas ayudan a desarrollar de manera eficiente herramientas de ML, estas ventajas también incluyen la tokenización del texto, la conversión de las palabras a vectores de características, y la organización de los datos en lotes para el entrenamiento. Además, Python proporciona herramientas para evaluar el rendimiento del modelo, como las métricas unitarias como precisión, recall y F-measure o métricas compuestas como ROUGE o BLEU.

4.2.10.6. Google Colab.

Google Colaboratory[18], conocido comúnmente como Google Colab, es una plataforma ofrecida de forma gratuita por Google que permite escribir y ejecutar código Python en el navegador. En concreto, permite ejecutar cuadernos Jupyter sin tener que preocuparse por el hardware o el software instalado en el ordenador.[69], es decir, ofrece un entorno de programación interactivo que no requiere configuración y proporciona acceso a recursos de computación gratuitos, incluyendo entornos con la utilización tanto de unidades de procesamiento gráfico (GPUs, por sus siglas en inglés) y unidades centrales de

[16] Pipeline: proceso mediante el cual se puede usar un modelo QA sin la necesidad de descargarlo
[17] Python: Conceptos y más sobre este lenguaje de programación
[18] GoogleColab: Fundamentos y cosas importantes

procesamiento (CPUs). Google Colab es utilizado para la investigación y el desarrollo en tareas de aprendizaje automático e inteligencia artificial, incluyendo modelos de Question Answering [70].

Google Colab permite escribir y ejecutar y compartir código de ejecución con extensiones como **ipynb** o **py,** además de poder recurrir a recursos computacionales como almacenamiento temporal o el mismo uso de las GPUs proporcionando una serie de bibliotecas preinstaladas que son útiles para el aprendizaje automático y el procesamiento del lenguaje natural, incluyendo TensorFlow, PyTorch, Keras[71] y otra más. También proporciona la facilidad de guardar y cargar notebooks directamente una cuenta de Google Drive, lo que facilita la gestión de los proyectos.

4.2.10.7. Recursos GPUs y CPUs.

En Google Colaboratory, una de las ventajas más importantes es que proporciona acceso a GPU que pueden acelerar significativamente los cálculos o cualquier otra acción que incluya procesamiento. Sin embargo, de forma predeterminada, Google Colab se ejecuta en CPU y, para usar GPU, se debe activar manualmente, a continuación, se mencionan cada una de estas dos formas de procesamiento de forma general.

- **GPUs:** Son dispositivos de hardware diseñados para manejar y acelerar cálculos y de procesamiento paralelo., además pueden realizar múltiples cálculos simultáneos, lo que permite distribuir los procesos de formación y puede acelerar considerablemente las operaciones de aprendizaje automático. Con las GPU, se pueden acumular muchos núcleos que utilizan menos recursos sin sacrificar la eficiencia o la potencia[72], siendo de utilidad para entrenar redes neuronales convolucionales o redesTransformers, en Google Colab la GPU más representativa y usada es la “T4 GPU” cuyo nombre completo es GPU NVIDIA Tesla T4, que proporciona una memoria RAM de 16 GB y una cantidad amplia de memoria temporal.
- **CPUs:** son las principales unidades de procesamiento de cualquier sistema computacional actual, responsables de ejecutar las instrucciones de un programa de ordenador, en otras palabras, son procesadores de propósito general capaces de gestionar una amplia gama de tareas, lo que las hace versátiles para diversas aplicaciones incluido el machine learning, aunque suelen ser menos eficientes que las GPUs para operaciones altamente paralelas, como el entrenamiento de modelos de IA.

4.2.10.8. TensorFlow y Pytorch.

Google Colab al trabajar con Python tiene la posibilidad de abordar dos bibliotecas como lo son TensorFlow y Pytorch, dos entornos utilizados para el desarrollo de herramientas de IA, a continuación, se detallan cada una de ellas.

- **TensorFlow[19]:** Es una plataforma de código abierto (open source) para el aprendizaje automático que utiliza gráficos de flujo de datos [73]. Los nodos del grafo representan operaciones matemáticas, mientras que los bordes del grafo representan las matrices de datos multidimensionales (tensores) que fluyen entre ellos. Se puede entrenar y ejecutar modelos de IA en GPU, CPU y TPU de distintas plataformas sin necesidad de reescribir el código, desde dispositivos portátiles a ordenadores de sobremesa o servidores de gama alta[74]. Además, ofrece varios flujos de trabajo con APIs de alto nivel disponibles para que personas principiantes y expertos creen modelos de ML en varios lenguajes (Java, Swift, C++, y Python)[75].
- **Pytorch**[20]: Es un marco de trabajo de código abierto (open source) desarrollado por Facebook's AI Research lab (FAIR)[76], con la finalidad de ser flexible y modular para la investigación, con la estabilidad y el apoyo necesarios para la implementación y creación de aplicaciones y sistemas de IA, realiza acciones de alto nivel como el cálculo y manejo de tensores[77], haciendo eficiente su uso al usar GPUs y CPUs. Sin embargo, suele ser deficiente en ambientes de alta producción y complejidad.

4.2.10.9. Métrica ROUGE.

Acrónimo de “Recall-Oriented Understudy for Gisting Evaluation”, ROUGE[21] es un conjunto de métricas y un paquete de software utilizado para evaluar sistemas o modelos de resumen automático, traducción automática y búsqueda de respuesta en el procesamiento del lenguaje natural. Las métricas comparan la salida de la modelo producida automáticamente con una información de referencia o un conjunto de referencias (producidos por humanos)[78]. ROUGE se basa en el cálculo de la superposición de n-gramas[22] entre el texto de respuesta generado por el modelo y el texto de referencia comúnmente obtenidos de los dataset(test), donde los n-gramas pueden ser palabras o secuencias de palabras dependiendo del valor de “n”. ROUGE es una métrica compuesta, es decir, se compone de diferentes submétricas que son:

[19] TensorFlow
[20] Pytorch
[21] ROUGE
[22] n-gramas significado y explicación

- **Precisión:** se refiere al número de predicciones positivas verdaderas dividido por el número total de predicciones positivas (es decir, el número de predicciones positivas verdaderas más el número de predicciones positivas falsas), a nivel general, mide el porcentaje de respuestas correctas generadas por un modelo entre todas las respuestas generadas[79].
- **Recall:** es una métrica que mide la frecuencia con la que un modelo de aprendizaje automático identifica correctamente instancias positivas (verdaderos positivos) a partir de todas las muestras positivas reales del conjunto de datos, en otras palabras, es la proporción de respuestas u observaciones positivas correctamente predichas respecto a las observaciones en la clase real[80].
- **Puntuación F1 score o F-measure:** Es la media armónica entre la precisión y la sensibilidad, generalmente es el valor que real de una predicción por parte de un modelo de IA[81].

ROUGE es una métrica que puede ser evaluada de diversas formas, las tres más usadas son: ROUGE-N, ROUGE-L y ROUGE-Lsum. En cada una de estas el cálculo de las submétricas (precisión, recall y F-measure) se realiza de diferente forma, a continuación, se presenta cada una de ellas.

- ***ROUGE-N:***

Se enfoca en la superposición de n-gramas, es decir analiza una o más palabras de respuesta generada por un sistema o modelo seleccionado y la compara en base a información de referencia, esta métrica a su vez se clasifica en varias subcategorías donde se destacan las siguientes: ROUGE-1 y ROUGE-2.

ROUGE-1: mide la coincidencia de unigramas (palabras de forma individual) entre el texto generado por el modelo y el texto de referencia, donde la precisión, la sensibilidad y la puntuación F1 son las submétricas que la conforman. A continuación, se presentan las fórmulas para realizar el cálculo de cada una de las submétricas en ROUGE-1.

$$\textbf{Precisión} = \frac{\text{Número de unigramas coincidentes}}{\text{total de unigramas generados}}$$

$$\textbf{Recall} = \frac{\text{Número de unigramas coincidentes}}{\text{total de unigramas en el texto de referencia}}$$

$$\textbf{F} - \textbf{measure} = \frac{2 * (\text{Precisión} * \text{Recall})}{\text{Precisión} + \text{Recall}}$$

ROUGE-2: Mide la coincidencia de bigramas (pares de palabras de forma consecutiva) entre el texto generado por el modelo y el texto de referencia, donde al igual que ROUGE-1, la precisión, la sensibilidad y la puntuación F1 son las submétricas que la conforman. A continuación, se presentan las fórmulas para realizar el cálculo de cada una de las submétricas en ROUGE-2.

$$\textbf{Precisión} = \frac{\text{Número de bigramas coincidentes}}{\text{total de bigramas generados}}$$

$$\textbf{Recall} = \frac{\text{Número de bigramas coincidentes}}{\text{total de bigramas en el texto de referencia}}$$

$$\textbf{F} - \textbf{measure} = \frac{2 * (\text{Precisión} * \text{Recall})}{\text{Precisión} + \text{Recall}}$$

- ***ROUGE-L.***

ROUGE-L, mide la coincidencia de la subsecuencia común más larga (sus siglas en inglés LCS) entre el texto generado por el modelo y el texto de referencia. La LCS, se debe evidenciar en ambos textos y sobre todo en el mismo orden, aunque puede o no estar de manera consecutiva[82], por lo que esta métrica es usada cotidianamente en comparar cualquier tipo de textos cortos como lo son las respuestas en modelos QA. A continuación, se presentan las fórmulas para realizar el cálculo de cada una de las submétricas en ROUGE-L.

$$\textbf{Precisión}: \frac{\text{LCS}}{\text{total de tokens en el texto generado}}$$

$$\textbf{Recall}: \frac{\text{LCS}}{\text{total de tokens en el texto de referencia}}$$

$$\textbf{F} - \textbf{measure}: \frac{2 * (\text{Precisión} * \text{Recall})}{\text{Precisión} + \text{Recall}}$$

- ***ROUGE-Lsum.***

Es una métrica similar a ROUUGE-L donde se mide la secuencia de texto más larga obtenida por un modelo respecto al texto de referencia, está enfocada específicamente a comparar estructuras de textos más completos tomando en cuenta los saltos de línea dentro de una oración por lo que está enfocada directamente en la comparación de resúmenes. Esta métrica se basa en la coincidencia de subsecuencias más largas (LCS) entre el resumen generado y el resumen de referencia, capturando así la estructura y el contenido de la oración.

$$\textbf{Precisión} = \frac{\text{LCS}}{\text{total de tokens en el texto generado}}$$

$$\textbf{Recall} = \frac{\text{LCS}}{\text{total de tokens en el texto de referencia}}$$

$$\textbf{F} - \textbf{measure} = \frac{2 * (\text{Precisión} * \text{Recall})}{\text{Precisión} + \text{Recall}}$$

4.2.10.10. Metodología CRISP-ML(Q)

La metodología CRISP-ML(Q)[23] siglas que significan Cross-Industry Standard Process for Machine Learning with Quality Assurance, es una extensión del modelo CRISP-DM (Cross-Industry Standard Process for Data Mining), diseñado específicamente para el desarrollo de aplicaciones de aprendizaje automático con un enfoque en la garantía de calidad, esta metodología estándar provee un marco de referencia y conjunto de buenas prácticas de acuerdo al proyecto en el que se aplique, se basa en un proceso iterativo que asegura que los proyectos de aprendizaje automático, incorpora principios de garantía de calidad en cada etapa del desarrollo, mitigando riesgos y asegurando que los modelos no solo sean precisos al momento de su creación, sino también sostenibles y efectivos a largo plazo[8].

CRISP-ML(Q) enfatiza la importancia de conocer y comprender los objetivos del negocio y los datos disponibles, integrando estas dos facetas en una sola fase inicial. Esto es fundamental ya que los objetivos de un proyecto y los datos es la materia prima de cualquier sistema de aprendizaje automático[83]. La estructura de CRISP-ML(Q) consta de seis fases iterativas, cada una con actividades o tareas de garantía de calidad, con la finalidad de asegurar que los modelos desarrollados sean robustos, precisos y enfocados en cumplir con los objetivos del negocio, estas etapas son:

1. Comprensión del negocio y los datos.
2. Ingeniería de los datos (preparación de datos)
3. Ingeniería del modelo de aprendizaje automático
4. Evaluación del modelo
5. Despliegue (implementación)
6. Mantenimiento.

En la **Figura 8** se puede visualizar las 3 seccione principales del ciclo de vida que enmarcan a las 6 fases correspondientes a la metodología CRISP-ML(Q).

[23] CRISP-ML(Q): metodología y marco de trabajo para proyectos ML

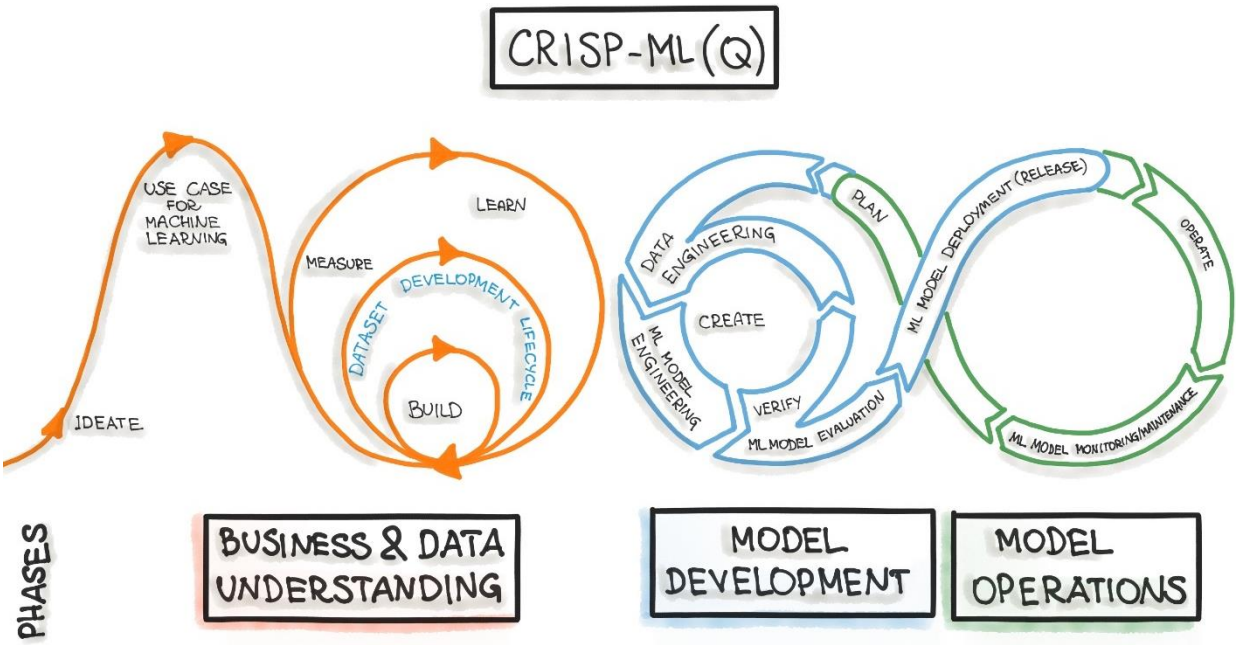


Figura 8. *Fases principales del CRISP-ML(Q): entendimiento de los datos y el negocio, desarrollo del modelo y operaciones del modelo. Esta figura es obtenida de la plataforma Ml-ops.org*[24]

- ***Fase 1: Comprensión del negocio y los datos.***

El proceso se centra en garantizar la viabilidad del proyecto de machine learning (ML) desde la gestación de la idea. Esta fase inicia con la delimitación del alcance, estableciendo criterios de éxito que abarcan tanto objetivos económicos (como la adquisición de herramientas) como las metas específicas que se pretenden alcanzar con el sistema de ML. Para asegurar que las metas sean cuantificables y factibles, es imperativo definir indicadores claves de desempeño (KPI, por sus siglas en inglés) como la precisión, el índice de errores, entre otros, que no solo son KPI, sino también métricas que evalúan la calidad de un modelo de ML[84].

Durante esta fase, es necesario especificar la disponibilidad de datos, restricciones regulatorias y requisitos de la aplicación del modelo de ML. La recopilación y verificación de la calidad de los datos al ser la base de los proyectos de ML es muy importante, dado que los datos guían el proceso de desarrollo del modelo. Esto implica documentar las propiedades estadísticas de los datos y sobre todo el proceso de generación de los mismos en caso de haberlos creado. Esto asegura que los datos utilizados sean adecuados y que el proyecto pueda cumplir con los objetivos comerciales establecidos, así como afirma Stonier Joann experta en datos e IA en Mastercard en la en un artículo de revista de IBM[25] “Los datos son el alimento de la IA. Si se desconoce la calidad de los datos, se desconoce la calidad de los resultados que se obtendrán de la IA”, aunque en los proyectos de IA y ML asegurar la calidad

[24] CRISP-ML(Q) en Ml-ops
[25] Revista de IBM

de los datos comprende un 60% de tiempo en estos proyectos, por lo que es necesario equilibrar el proceso de un proyecto o investigación respecto al tiempo que se dispone[85]. Para ello, algunas de las actividades que se pueden realizar en esta fase son: definir objetivos comerciales, traducir objetivos comerciales en objetivos de ML[86], recopilar y verificar datos, evaluar la viabilidad del proyecto y crear pruebas de concepto (POC)[26].

- ***Fase 2: Ingeniería de datos (preparación de datos).***

Esta segunda fase tiene como objetivo preparar datos para la siguiente fase, que es el modelado. Durante esta etapa se realizan tareas de selección de datos, limpieza de datos, ingeniería de características y estandarización de datos. También se aborda el problema de las clases desequilibradas aplicando técnicas como: ajuste de parámetros, muestras artificiales o el sobremuestreo y submuestreo.

La finalidad de la limpieza es corregir errores en los datos para no tener inconvenientes en el entrenamiento y construcción del modelo, de esa forma obtener un mejor resultado, en los casos en que el modelo de ML se enfoca en áreas específicas es necesario realizar actividades de aumento de datos (enfoques como Few-shot learning y la paráfrasis), y finalmente la normalización para mitigar el sesgo y discriminación de los datos. Algunas de las tareas o actividades dentro de esta fase incluyen: selección de datos, equilibrio de clases, limpieza de datos (reducción de ruido, imputación de datos), construcción de datos (ingeniería de funciones), aumento de datos, estandarización de datos.

- ***Fase 3: Ingeniería de modelos.***

Esta etapa se enfoca en el desarrollo de uno o varios modelos de ML. Aquí se enfoca en las limitaciones y requisitos establecidos en la primera fase. Se selecciona el o los modelos para cumplir con los objetivos, además aquí es donde se realiza el proceso de entrenamiento, considerando aspectos importantes como: rendimiento, equidad, escalabilidad y la demanda de recursos[87]. La importancia de cada métrica se ajusta según el caso de uso específico analizado en etapas anteriores, con el objetivo de asegurar que el modelo cumpla con los objetivos y restricciones del proyecto.

Los proyectos de ML para demostrar validez deben ser reproducibles. Por lo que para abordar esto, es primordial: recopilar y documentar los metadatos usados en el entrenamiento, incluyendo el algoritmo utilizado (en caso de ajustar un modelo se debe documentarlo), los conjuntos de datos de entrenamiento(train), validación(validation) y prueba(test), los hiperparámetros (batch_size, epochs, learning_rate, entre otros) y la descripción del entorno de ejecución.

[26] Prueba de concepto (POC)

- ***Fase 4: Evaluación del modelo.***

Esta es la fase posterior al entrenamiento y se enfoca en validar el rendimiento del modelo entrenado utilizando el conjunto de pruebas más conocido como test. Esta etapa verificar que el modelo funcione correctamente antes de una posible implementación en un entorno real. Además, se verifica la robustez del modelo con datos diferentes a los del dataset de entrenamiento, siempre y cuando estén en el mismo contexto, y así evidenciar el comportamiento del modelo de ML como entornos de prueba. Esta fase al pertenecer a CRISP-ML(Q) se debe realizar la respectiva documentación donde se detalle los resultados obtenidos durante la evaluación para garantizar la transparencia y trazabilidad en el proceso de desarrollo del modelo. Esta documentación incluye los métodos utilizados y las métricas de evaluación, un ejemplo en un ambiente de los modelos de ML es la métrica ROUGE.

- ***Fase 5: Implementación o despliegue.***

Aquí se integra el modelo de ML a un sistema nuevo o existentes, ya en un ambiente de producción. Además, esta fase abarca la definición del hardware necesario, la evaluación del modelo en un entorno de producción (como pruebas A/B), la aceptación del usuario, pruebas de usabilidad, entre otras.

- ***Fase 6: Mantenimiento.***

Esta fase incluye la evaluación continua del modelo para problemas irregulares, por ejemplo, detectar caídas en el rendimiento, esto se puede dar debido a cambios en los datos en el entorno real. Además, el rendimiento del modelo puede verse afectado por el hardware y la infraestructura de usada durante la implementación. Por lo tanto, es crucial un monitoreo constante para decidir si es necesario realizar cambios en el hardware o en el modelo de ML.

En la **Tabla 3** se resumen las tareas que se pueden realizar en cada fase del modelo CRISP-ML(Q), destacando que se deben ejecutar dependiendo del enfoque y proyecto realizado.

Tabla 3*. Tareas que se pueden realizar en cada fase de CRISP-ML(Q)*

Tareas de cada fase de CRISP-ML(Q)	
Fase	**Tareas**
Comprensión del negocio y los datos.	• Traducir objetivos del negocio en objetivos de ML • Recopilar y verificar datos • Crear POC
Ingeniería de los datos (preparación de datos)	• Selección de datos y funciones • Equilibrio de clases • Limpieza de datos (reducción de ruido, imputación de datos) • Ingeniería de funciones (construcción de datos) • Aumento de datos • Estandarización de datos

Fase	Tareas
Ingeniería del modelo de aprendizaje automático	• Selección del modelo y compresión del mismo • Agregar conocimiento del dominio para especializar el modelo • Aplicar técnicas como: Transfer Learning o Fine-Tuning • Entrenamiento del modelo • Documentar el modelo de ML y experimentos
Evaluación del modelo	• Validar el l modelo mediante métricas (ROUGE u otra) • Determinar la solidez mediante pruebas • Aumentar la explicabilidad del modelo • Documentar la fase de evaluación
Despliegue (implementación)	• Evaluar el modelo en condiciones de producción • Asegurar la aceptación y usabilidad del usuario
Mantenimiento	• Supervisar la eficiencia y eficacia del servicio de predicción del modelo • Comparar con los criterios de éxito especificados previamente (umbrales) • Volver a entrenar el modelo si es necesario

Nota: Si se usa esta metodología como un marco de referencia no es necesario cumplir con todas las actividades, se debe seleccionar las que aporten al proyecto, sistema o trabajo en cuestión.

4.2.11. Trabajos Relacionados

Basado en la revisión bibliográfica realizada durante la investigación, se recopilaron diversos artículos, papers, libros y otras fuentes de información que respaldan este trabajo. En la **Tabla 4**, se detallan algunos de los trabajos fundamentales para el desarrollo del TIC. Se incluye el identificador del documento, el título, la referencia en formato IEEE y un resumen que describe su aporte e importancia en la elaboración del TIC.

Tabla 4. *Estudios principales seleccionados*

Trabajos Relacionados			
Indicador	**Título**	**Referencia**	**Resumen**
T1	Fine-Tuned Transformer Models for Question Answering	[38]	Este trabajo se enfoca en proponer una alternativa a las redes neuronales convolucionales en la obtención de respuestas, elaborando un modelo basado en las redes Transfomers para cumplir con esta tarea de manera más eficiente. Esta investigación es primordial para el desarrollo de este TIC ya que proporciona una visión adecuada sobre el tipo de red que se utiliza en la elaboración de modelos Question Answering enfocado en obtener respuestas en determinado contexto.
T2	Attention Is All You Need	[40]	Este artículo se enfoca en demostrar la eficiencia de las redes neuronales Transformers para el PLN, trabajando específicamente con textos, ya sea en tareas de traducción o QA, por lo que este artículo tuvo un gran aporte al TIC, ya sirvió de guía para seleccionar la red neuronal que se utilizó en la investigación, además proporcionó información sobre la eficiencia de esta red en el PLN trabajando con dataset de entrenamiento, validación y test grandes o también con dataset limitados en cuento a cantidad de datos.

Indicador	Título	Referencia	Resumen
T3	A Dataset of Information-Seeking Questions and answers Anchored in Research Papers	[3]	Este artículo aborda la necesidad plena de abarcar un dataset específico para entrenar modelos QA enfocados en algo particular, obteniendo preguntas relevantes, logrando con ello aumentar el valor de forma positiva en las métricas de rendimiento a nivel de entrenamiento y evaluación de los QA. Este trabajo es esencial para comprender la necesidad de crear un dataset en el contexto del TIC.
T4	Chatbot basado en una versión ligera del modelo BERT para resolver inquietudes relacionadas con matrículas y homologaciones en la Universidad Nacional de Loja	[6]	En este trabajo se muestra la construcción de un chatbot basado en el modelo DistilBERT basado en la arquitectura Transformer para responder a preguntas sobre un contexto específico en la UNL, el autor de este artículo realiza un proceso desde la concepción de la idea hasta su evaluación. Así mismo, como muestra de un alto rendimiento se logra obtener un porcentaje de precisión de 80.66%, esto ayuda en gran medida a la confianza en cuanto a las respuestas que brinda el chattBot. Este artículo ayudó a seleccionar el modelo pre-entrenado DistilBERT para enfocarlo en un contexto específico logrando mantener un alto rendimiento y eficiencia.
T5	The DistilBERT Model: A Promising Approach to Improve Machine Reading Comprehension Models.	[88]	En este artículo se analiza la comprensión de lectura automática (MRC) a través del modelo derivado de BERT llamado DistilBERT, menciona los desafíos a los que se somete para extraer respuestas en base a la entrada (una pregunta) y compara con algunos modelos de referencia tales como: GPT-3, RoBERTa, BERT, entre otros; evaluados en dos parámetros que son: F1(métrica que combina la precisión y el recall del modelo) y EM respecto a diferentes DataSet usados para estas tareas, como lo son: SQuAD1.0, SQuAD2.0, TriviaQA. CoQA, entre otros, con ello nos da algunas perspectivas de los parámetros (número de capas, función de activación y estructura general).
T6	DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter	[25]	Este artículo propone un enfoque innovador para mejorar la eficiencia de modelos de procesamiento de lenguaje natural mediante transferencia de aprendizaje desde modelos pre-entrenados a gran escala, como BERT. Se introduce DistilBERT, un modelo de representación de lenguaje general más pequeño y rápido. La relevancia de este trabajo en la elaboración de este TIC radica en su capacidad para pre-entrenar modelos más eficientes y ligeros, adecuados para ejecución en dispositivos con recursos computacionales limitados. Esta aproximación no solo optimiza los recursos de entrenamiento y procesamiento, sino que también demuestra la capacidad para ajustar DistilBERT a contextos específicos.

Indicador	Título	Referencia	Resumen
T7	Improving Question Answering Performance Using Knowledge Distillation and Active Learning	[89]	En este artículo se menciona estrategias de Aprendizaje Automático para mejorar un modelo QA utilizando una pequeña proporción de datos, específicamente el 20% de SQuAD en el dataset de entrenamiento, por lo que este trabajo es indispensable para fortalecer el conocimiento y aumentar las métricas y los resultados del modelo QA planteado en el presente TIC.
T8	Agente virtual para brindar asistencia acerca del covid-19	[7]	En este artículo se evidencia la implementación de la técnica Fine-Tuning, entrenando un modelo Question Answering con el dataset SQuAD2.0 para responder a preguntas sobre el Covid-19, además se utilizan GPUs como forma procesamiento durante el entrenamiento del modelo, al tratarse de un trabajo similar al que se está realizando, este documento presenta una guía para desarrollar este proyecto.
T9	Fine-Tuned Transformer Models for Question Answering	[90]	Este artículo habla sobre los hiperparámetros más sobresalientes en el ajuste de un modelo QA además utiliza la red neuronal Transformer para procesar los datos obtenidos en un dataset de QA, por lo que este documento ayuda como base para encontrar los puntos y valores adecuados durante la realización de la técnica Fine-Tuning en el modelo QA para responder a preguntas sobre el contenido académico del presente TIC.
T10	Training tips for the transformer model	[91]	Este artículo detalla experimentos en traducción automática neuronal utilizando Tensor2Tensor y el modelo Transformer, explorando parámetros críticos como la gestión de memoria, la estabilidad del entrenamiento y el tiempo necesario. Su contribución principal a este TIC radica en proporcionar recomendaciones prácticas para optimizar el entrenamiento, incluyendo estrategias para manejar conjuntos de datos y modelos complejos en entornos de GPU. Estas sugerencias que se dan en este trabajo son invaluablemente transferibles a la construcción del modelo Question Answering, donde la gestión eficiente de recursos y la optimización de parámetros como el tamaño del lote y la tasa de aprendizaje son cruciales para mejorar la precisión y la eficiencia del modelo.
T11	Evaluating Question Answering Evaluation	[92]	En este artículo se menciona la forma de implementar la métrica ROUGE sobre los modelos de realización de resúmenes y los modelos QA, habla de las herramientas como las librerías de Python para implementar de forma automática esta métrica, por lo que es esencial en este trabajo debido a la importancia que este radica para poder cumplir el objetivo 2 del presente trabajo.

Indicador	Título	Referencia	Resumen
T12	The limits of automatic summarization according to ROUGE	[5]	Este artículo investiga las dificultades clave de construcción y la evaluación mediante la métrica ROUGE, demostrando la efectividad empírica de los algoritmos(modelos) en varios conjuntos de datos. Su aporte crucial para la construcción de este TIC radica en proporcionar perspectivas sobre la selección de métricas y estrategias de evaluación, esenciales para mejorar la precisión y utilidad de un modelo, y sobre todo muestra algunas limitaciones de la evaluación de un modelo.
T13	Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology	[8]	Este trabajo propone la metodología CRISP-ML(Q) para la elaboración de proyectos de IA especialmente de ML. Una metodología que puede ser ocupada como framework y que propone documentación como método de calidad. Este artículo es indispensable ya que a través de él se establece un marco de referencia que guía el proceso de construcción de este TIC para el ajuste de un modelo QA basado en DistilBERT y su posterior evaluación.

4.2.11.1. Conclusión sobre los trabajos relacionados

Los trabajos relacionados desempeñaron un papel crucial en la construcción y evaluación del modelo QA basado en DistilBERT. En primer lugar, se demostró la eficacia de los modelos de lenguaje pre-entrenados como DistilBERT para aplicaciones de procesamiento de lenguaje natural (PLN), destacando su capacidad para mejorar significativamente la precisión y eficiencia en tareas de PLN en contextos específicos (T1, T2, T4, T5). Además, estos estudios subrayaron la importancia de utilizar datasets especializados como lo es SQuAD para entrenar y evaluar modelos QA, asegurando así la representatividad y relevancia de los datos utilizados (T3).

En términos de herramientas y técnicas, los trabajos exploraron estrategias avanzadas como el ajuste de hiperparámetros y la utilización de métricas de evaluación robustas como ROUGE, fundamentales para optimizar y validar el rendimiento del modelo QA basado en DistilBERT en diversas condiciones (T6, T7, T8, T10). Estas investigaciones también proporcionaron directrices prácticas para la implementación de metodologías especializadas para elaborar proyectos de Machine Learning como lo es CRISP-ML(Q)(T13), que estructuraron el proceso de desarrollo desde la definición del problema hasta la evaluación continua del modelo en entornos dinámicos y exigentes (T9, T12).

En conjunto, estos estudios respaldaron la selección estratégica y el ajuste efectivo de DistilBERT para el enfoque Question Answering, estableciendo un marco metodológico sólido y proporcionando herramientas prácticas para garantizar la eficiencia, precisión y mantenimiento a largo plazo del modelo.

5. Metodología

5.1. Área de estudio

El presente trabajo de Integración Curricular se desarrolló en la Universidad Nacional de Loja, ubicada en la ciudad de Loja, Ecuador. La investigación se llevó a cabo específicamente en la Facultad de Energía, las Industrias y los Recursos Naturales no Renovables (FEIRNNR), carrera de Computación en el itinerario “Sistemas Inteligentes” enfocado en el Aprendizaje Automático específicamente en la construcción de un modelo Question Answering. Esta área de estudio fue elegida debido a la relevancia y necesidad de ajustar e implementar un modelo QA (DistilBERT), para responder preguntas basadas en el contenido extraído de tareas académicas de los estudiantes en la carrera antes mencionada. En la **Figura 9** se muestra la localización de la Carrera de Computación de la UNL.

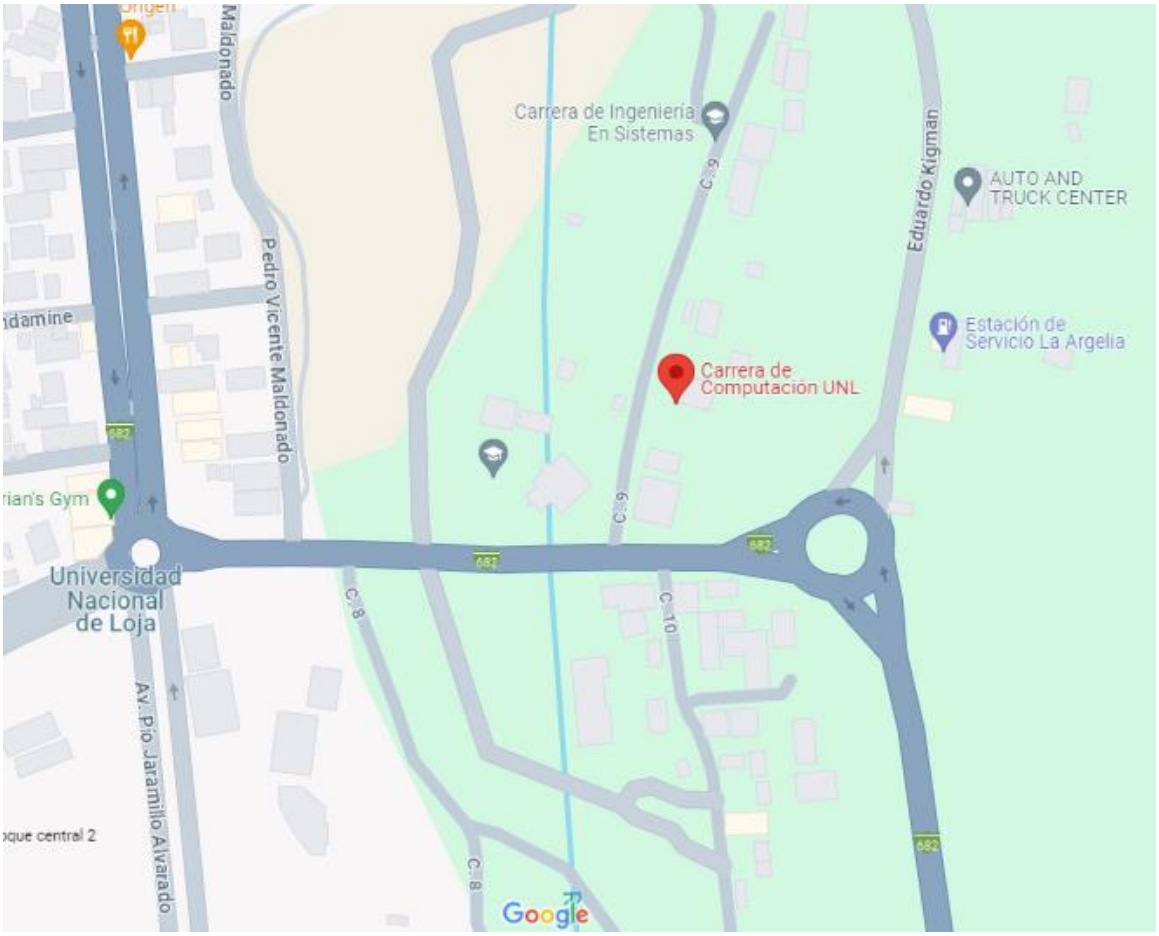


Figura 9. *Ubicación de la carrera de Computación en la UNL. Para mejor visualización acceda desde google maps*

5.2. Procedimiento

Para guiar el proceso de ajuste del modelo QA se empleó las siguientes técnicas, e instrumentos: entrevista, el crowdsourcing, el enfoque Few-shot Learning, la paráfrasis y el método científico (investigación cuantitativa experimental). Para la evaluación se implementó la métrica ROUGE y la técnica A/B Testing. Además, se utilizó la metodología CRISP-ML(Q) como un marco de referencia, el cual proporcionó una estructura sistemática para llevar a cabo el desarrollo del modelo Question Answering. Específicamente, se emplearon las primeras cuatro fases del ciclo de vida de CRISP-ML(Q): comprensión de datos y negocios, ingeniería de datos (preparación de datos) y la ingeniería de modelos de aprendizaje automático para dar cumplimiento al primer objetivo; y la evaluación de modelos de aprendizaje automático para cumplir el segundo objetivo.

5.2.1. Método científico

Para el desarrollo del TIC, se empleó la investigación cuantitativa experimental. Este enfoque se fundamenta principalmente en la definición de objetivos y preguntas de investigación cuantificables, además de la revisión de la literatura existente, ya que se busca verificar el valor de la métrica ROUGE al modificar los hiperparámetros (epochs, batch_zise, learning-rate, entre otras) del modelo DistilBERT. Mediante este proceso, se recolectaron y analizaron datos, manipularon variables e hiperparámetros, y se realizaron experimentos con el propósito de responder a la pregunta cuantitativa: “¿Qué puntuación ROUGE se obtiene al ajustar un modelo QA implementando la técnica Fine-Tuning en DistilBERT para dar respuesta a preguntas sobre el contenido extraído de tareas académicas de la carrera de Computación en la UNL?”.

El enfoque cuantitativo experimental permitió utilizar el modelo DistilBERT y la estructura de datos SQuAD1.0, para ajustar de manera cuantitativa el rendimiento del modelo en la tarea de Question Answering respecto al dominio específico de las tareas académicas de la carrera de Computación en la UNL y para la evaluación se utilizó la métrica ROUGE. Además, se llevaron a cabo investigaciones bibliográficas y de campo. Las investigaciones bibliográficas (vea **Trabajos Relacionados**) permitieron recolectar estudios relacionados con las características y el funcionamiento de los modelos basados DistilBERT, y su aplicación en tareas de Question Answering. Por otro lado, la investigación de campo incluyó una entrevista con el director del proyecto de vinculación de la carrera de Computación (véase **Anexo 2**), lo cual permitió verificar la necesidad de un modelo QA en este contexto y justificar la viabilidad y relevancia del proyecto, además de la búsqueda de las tareas académicas para crear el dataset.

5.2.2. Objetivo 1: Adaptar el modelo DistilBERT para responder a preguntas sobre el contenido extraído de tareas académicas en la carrera de Computación de la UNL

5.2.2.1. Comprensión de datos y negocios

En esta fase, se inició con la comprensión del proyecto de vinculación de la carrera de Computación llamado "Inclusión lectora de estudiantes con discapacidad visual mediante innovación tecnológica" y a su vez se realizó una revisión de sus requisitos (ver **Anexo *1***), para estudiar el contexto de este proyecto se procedió a realizar una entrevista[27](ver **Anexo *2***) al Ing. Oscar Cumbicus director del proyecto, misma que permitió obtener información sobre la necesidad de un modelo QA para responder a preguntas sobre el contenido extraído de tareas académicas en la Carrera de Computación (ver **Anexo *2*** sección Componentes). Para realizar la entrevista se agendó una reunión online en la plataforma Zoom[28] con el director del proyecto el cual ayudó grabando la entrevista y también su posterior publicación en YouTube[29], el intervalo exacto donde se realizó esta entrevista es entre los 16min: 18s y 39min: 35s respectivamente. También se llevó a cabo una búsqueda de información tradicional en la biblioteca virtual de la UNL y en bases de datos e indexadores como: Scopus, ACM, Springer, ACL anthology, entre otras (ver **Anexo *3***) esto permitió obtener estudios relevantes (ver sección **Trabajos Relacionados**) y establecer las bases de conocimiento necesarias para el desarrollo del proyecto. Finalmente, mediante la plataforma de Google Docs y la extensión de Google "YouTube Summary with ChatGPT & Claude[30]" se procedió a documentar y trascribir las respuestas de dicha entrevista.

5.2.2.2. Ingeniería de datos (preparación de datos).

En esta fase de la metodología CRISP-ML(Q) se realizó 3 tareas indispensables, las cuales son: recopilar documentos de tareas académicas, construcción del dataset en formato SQUAD1.0 y particionar los datos en entrenamiento (train) y validación(test).

Tarea 1: Recopilar documentos de tareas académicas.

En esta etapa, se recopiló tareas académicas (en formato PDF) al azar en diferentes materias cursadas durante el periodo 2021-2024 en la carrera de Computación de distintos ciclos académicos, los documentos obtenidos se almacenaron un repositorio de acceso abierto en drive[31]. Para aquellas tareas que no se encontraban en formato pdf, se llevó a cabo la conversión necesaria mediante herramientas gratuitas como lo es iLovepdf[32]. De esta

[27] Requisitos del proyecto en el repositorio en drive
[28] zoom: plataforma para realizar reuniones online
[29] Entrevista en Youtube, o en drive
[30] YouTube Summary with ChatGPT & Claude
[31] Repositorio de documentos recolectados por materia
[32] Herramienta iLovepdf para conversión de documentos

manera, se logró obtener documentos con las mismas características y con mayor facilidad de procesamiento para las siguientes fases de CRISP-ML(Q).

Tarea 2: Construcción del dataset en formato SQUAD1.0

En esta tarea existen 2 actividades, las cuales son: aplicar la técnica crowdsourcing y crear el dataset en formato SQuAD1.0.

- ***Aplicar la técnica crowdsourcing****:* En esta actividad, se empleó la técnica de crowdsourcing para generar un conjunto de datos de preguntas relacionadas con los documentos académicos recolectados en la actividad anterior (Recopilar documentos de tareas académicas). En esta técnica, al implicar la colaboración de diversas personas, se optó por solicitar el apoyo de manera informal mediante mensajes simples en redes sociales (WhatsApp y Facebook) a estudiantes de la carrera de Computación de la Universidad Nacional de Loja, para crear preguntas basadas en el contenido de las tareas académicas. Para ello se creó un documento de acceso abierto en Excel en un repositorio de drive[33], una plataforma en línea para facilitar la colaboración en la generación de preguntas. El documento contiene columnas específicas para que los estudiantes pudieran registrar el nombre del documento seleccionado, la pregunta generada y otros atributos opcionales. Además, se planteó 11 instrucciones dentro del documento para las personas que colaboraron, de las cuales se destacan: Generar la cantidad de preguntas que se deseen y no editar secciones que no hayan escrito ellos mismos. Finalmente, para asegurar la calidad de las preguntas se comprobó de forma manual que estas dispongan de los elementos esenciales en una pregunta formulada correctamente: interrogante, verbo y complemento. En el caso que no disponían con todos los elementos se procedió a realizar ajustes o en otros casos se procedió a suprimir dichas preguntas mal planteadas.
    - ***Creación del dataset en formato SQuAD1.0****:* En esta actividad, se procedió a crear un dataset en formato SQuAD 1.0 a partir de preguntas recopiladas mediante crowdsourcing en una hoja de cálculo en Excel. El proceso de ampliación de datos se realizó en dos subactividades principales: la aplicación del enfoque Few-shot Learning en la que se utilizó la herramienta de IA generativa ChatGPT para crear alrededor de cinco variantes de cada pregunta original y la aplicación de la técnica de paráfrasis donde a cada variante generada se sometió a un proceso en el que se empleó sinónimos y reestructuraciones sintácticas, para obtener una pregunta adicional por cada pregunta generada mediante Few-shot Learning. Esta estrategia de aumento de datos permitió expandir significativamente el conjunto inicial, generando aproximadamente diez preguntas derivadas de cada pregunta original. Como resultado, se alcanzó un total de 800 entradas generadas. Finalmente, con la participación directa del autor, el dataset se

[33] Repositorio en Drive con la técnica crowdsourcing

completó hasta alcanzar 1410 entradas en total. Luego, se procedió a completar los atributos (id, title, context y answers) del formato SQuAD 1.0 en base a los documentos de los cuales se obtuvieron las preguntas. En cada una de las variables se utilizó criterios de inclusión y exclusión para su definición (véase **Tabla 8**). El primer valor de la variable "id" se asignó al azar, asegurándose de que fuera un dato alfanumérico, y los valores siguientes se generaron en secuencia. Para la variable "title", se utilizaron conjuntos de palabras que representaban el contexto de la materia abordada por el documento. En la variable "context", se incluyeron las frases o párrafos donde se encontraba la respuesta a la pregunta, misma que ya se colocó en la variable "question". En la variable "answers", se introdujo un diccionario[34] con dos valores: el texto de respuesta, representado por "text", y el índice de inicio de inicio de la respuesta dentro del contexto, representado por "answer_start". Para determinar este último valor, se utilizó la herramienta online de acceso abierto "contadordepalabras[35]". Este dataset se organizó en un archivo Excel alojado en el repositorio de Google Drive[36]. Además, se implementó un código sencillo en Python con la función "find" para verificar que el índice encontrado por la app sea el correcto, esto únicamente en ocasiones en las que se consideró fuese necesario. En la **Figura 10** se puede visualizar un ejemplo de la verificación en la que se comparó el índice de respuesta proporcionado por la aplicación web "contadordepalabras" y utilizando la función antes mencionada.

```
context = """Objetivo general: Evaluar la seguridad de una aplicación web."""
answer = "Evaluar la seguridad de una aplicación web"
answer_start = context.find(answer)
print(answer_start)
indiceApp=18
if answer_start==indiceApp:
  print("Son iguales")
```

```
18
Son iguales
```

Figura 10. *Código de verificación del índice de inicio de la respuesta*

Tarea 3: Particionar los datos en entrenamiento y validación.

En esta tarea se procedió a convertir el dataset que se creó en la tarea anterior en formato ".csv" a formato JSON requerido por SQuAD1.0 mediante la librería "json" de Python, que automatizó la conversión de los datos. Posteriormente, se realizó una revisión para proceder a guardar el archivo en formato JSON en el repositorio[37]. A continuación, con la ayuda de la herramienta de python *test_split* se dividió en conjuntos de entrenamiento(train) y evaluación(test), utilizando una proporción de 90% de los datos destinados para el *train* y el

[34] Diccionario: una variable que almacena más de un único valor
[35] Herramienta contadordepalabras
[36] dataset en formato .csv
[37] Dataset en formato json

10% restante para el *test*, luego de la primera división se procedió a realizar una subdivisión del conjunto de datos *train* correspondiente a un 75% para *train* y el 25% restante para validación(validation). Finalmente, se obtuvieron los datos correspondientes para cada sección y se almacenaron en github[38], teniendo en cuenta que al final de este proceso se obtuvieron 3 conjuntos de datos: train, validation y test. En la **Figura 11** se puede observar el proceso de división del dataset.

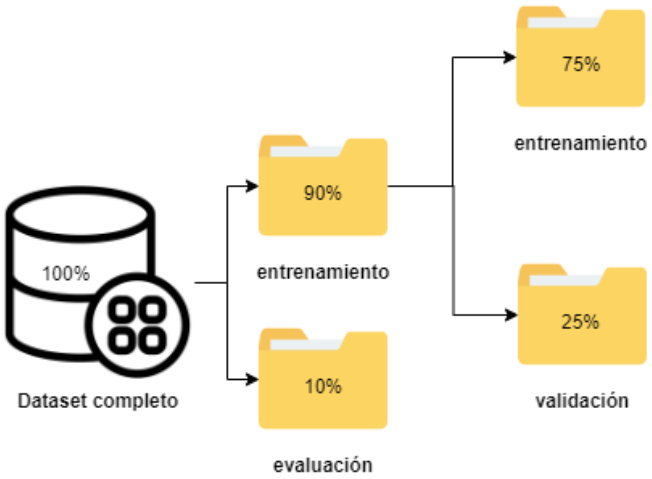


Figura 11. *Proceso de división del dataset en: train, validatión y test*

5.2.2.3. Ingeniería de modelos de aprendizaje automático.

Durante esta etapa, se procedió a cargar los datos de entrenamiento (train , validation) y evaluación (test) desde el repositorio mencionado en la fase anterior, para continuar a la configuración de las credenciales del repositorio de Hugging Face(como un paso adicional se generó una clave de escritura) para guardar el modelo generado después del entrenamiento. Luego, se importó el tokenizador (AutoTokenizer) y el modelo pre-entrenado base DistilBERT (distilbert-uncased)[39], al cual se le aplicaron las técnicas de Transfer Learning y en especial Fine-Tuning.

Para implementar la técnica de Fine-Tuning en el modelo de Question Answering, se realizaron varios experimentos ajustando diferentes hiperparámetros del modelo. Los valores ajustados incluyeron: tasa de aprendizaje (learning rate), épocas (epochs), tamaño del lote (batch_size), semilla (seed), optimizador (optimizer), weight decay (se puede traducir como decaimiento del peso) y la longitud máxima de entrada (max_length). Además, se empleó la primera versión de SQuAD para el dataset específico creado en la fase anterior. Se llevaron a cabo cuatro experimentos, cada uno con variaciones en los hiperparámetros mencionados,

[38] Dataset de train y test
[39] DistilBERT

mediante el uso del método "Train" de PyTorch, se entrenó el modelo QA en cada experimento, asegurando la consistencia mediante el uso del mismo valor de semilla para trabajar con los mismos datos en cada uno de los experimentos. El proceso de entrenamiento duró aproximadamente entre 50 y 70 minutos por cada modelo, empleando el entorno de ejecución GPU T4 en Google Colab. Durante el entrenamiento, se registraron y compararon varios valores y estadísticas obtenidas luego del entrenamiento para evaluar el rendimiento de los modelos, estas incluyen:

- Pérdida durante el entrenamiento
- Precisión de los logits finales en el entrenamiento
- Precisión de los logits de inicio en el entrenamiento
- Pérdida durante la validación
- Precisión de los logits finales en la validación
- Precisión de los logits de inicio en la validación
- Época
- Nivel de extracción F1 score
- Tiempo aproximado de entrenamiento (min)

Estas métricas permitieron seleccionar el modelo con las mejores puntuaciones generales, el cual fue desplegado en la plataforma de acceso abierto Hugging Face. Para garantizar la reproducibilidad y transparencia del proceso, se generó un TensorBoard con los datos de entrenamiento, registrando la curva de aprendizaje, pérdida y otros valores relevantes comprendidos en gráficas. Esto se realizó para asegurar que, en caso de error al desplegar el modelo en Hugging Face, no se perdieran estas valiosas gráficas, ya que el almacenamiento en el entorno temporal de Google Colab podría no preservarlas en su versión gratuita misma que se utilizó en este TIC. Además, todo el código y los parámetros utilizados del modelo con las estadísticas más sobresalientes de los cuatro experimentos, se documentaron en el archivo "ModeloQA_TIC1.ipynb". Este archivo, junto con los otros tres archivos de los experimentos, está disponible en un repositorio de GitHub[40], proporcionando todos los detalles necesarios para replicar el entrenamiento del modelo, desde la carga de datos, limpieza de datos, división de datos, el ajuste de hiperparámetros y el proceso para elaborar la evaluación en la siguiente fase de la presente metodología.

[40] Código en github

5.2.3. Objetivo 2: Evaluar el rendimiento del modelo mediante el análisis de la métrica ROUGE en la obtención de respuestas dadas por el modelo QA.

Este objetivo se centra en la fase de evaluación del modelo, siguiendo la metodología CRISP-ML(Q). La evaluación se realizó utilizando el dataset de evaluación (test) del modelo con las mejores prestaciones con la finalidad de medir cuantitativamente su rendimiento, este ofrece tres opciones de acceso: mediante el archivo en formato h5, a través del pipeline, o directamente desde la plataforma Hugging Face donde fue desplegado. Considerando las limitaciones de uso desde Hugging Face (opciones de pago para poder utilizar todas las funcionalidades), se optó por utilizar el modelo en formato **h5** para su implementación de forma local y el pipeline, esto debido a su accesibilidad y facilidad de implementación. Esta elección permite proceder con la evaluación de manera eficiente y consistente.

5.2.3.1. Evaluación de modelos de aprendizaje automático

En esta fase final, se llevó a cabo la evaluación del modelo de QA utilizando la métrica ROUGE. Esta métrica al contar con 3 submétricas internas (precisión, recall y F-measure) ayuda a cuantificar la calidad de las respuestas generadas por el modelo, comparando las respuestas generadas por el modelo respecto a las respuestas de referencia proporcionadas en el dataset “test” que consta de 141 datos, esta evaluación se la realizó desde dos perspectivas diferentes: una evaluación automática con la función rouge_metric de python de forma unitaria y general, y mediante la aplicación de la técnica A/B testing entre el valor F-measure de ROUGE-L (valor elegido debido a que mantiene un rendimiento medio entre todas las métricas ROUGE) y las calificaciones dadas por el experto en procesamiento de lenguaje natural el Ing. Oscar Cumbicus Pineda en un rango de valores entre 0 a 100. En las dos perspectivas el primer paso que se realizó fue obtener las respuestas de referencia del dataset *test* para poder realizar los cálculos.

- ***Medición de ROUGE de forma automática***

La evaluación automática se realizó utilizando dos conjuntos de códigos para obtener las respuestas del modelo y del conjunto respuestas de referencia. Mediante la biblioteca rouge_score, se compararon individualmente las respuestas de referencia con las respuestas generadas por el modelo para calcular los valores de ROUGE. Además, se elaboró un código para calcular las métricas ROUGE de forma general para los 141 datos del conjunto de datos "test", los resultados de la evaluación automática se presentaron en una tabla que muestra los valores obtenidos para ROUGE-1, ROUGE-2, ROUGE-L y ROUGE-LSum, incluyendo las submétricas: precisión, recall y F-measure en niveles bajos, medios y altos. Los valores F-measure bajos, medios y picos se destacan para cada métrica, proporcionando una visión clara de la efectividad y consistencia del modelo en la generación de respuestas, lo que ayuda

a proporcionar una comprensión exhaustiva del rendimiento del modelo de Question Answering.

- ***A/B Testing***

Para evaluar la efectividad de la métrica ROUGE en comparación con las calificaciones otorgadas por un experto en Procesamiento del Lenguaje Natural (PLN), se realizó un A/B Testing sobre una muestra ponderada del 10% del conjunto de datos de evaluación, equivalente a 14 datos seleccionados aleatoriamente, para asegurar una representación equilibrada de la diversidad del dataset. El procedimiento de evaluación se llevó a cabo mediante dos procesos. Primero, se obtuvo el valor medio de F-measure de la métrica ROUGE-L obtenido en la evaluación automática, valores se establecieron de 0 a 100 para facilitar la comparación directa con las calificaciones del experto. Luego, el experto en PLN evaluó cada respuesta generada, otorgando una calificación en la misma escala de 0 a 100. Para analizar los resultados, se compararon los valores de F-measure de ROUGE-L (sujeto A) y las calificaciones del experto (sujeto B) para cada una de las 14 preguntas. Se calcularon los promedios de las calificaciones otorgadas por ambos métodos, para obtener una perspectiva real sobre la eficacia del modelo. Finalmente, el profesional del PLN validó el modelo de acuerdo a las respuestas y resultados obtenidos durante el A/B Testing.

5.2.4. Instrumentos utilizados

5.2.4.1. Entrevista

Se realizó una entrevista al director del proyecto para comprender las necesidades y el contexto del proyecto de vinculación. La entrevista se llevó a cabo a través de la plataforma Zoom en una reunión programada. Se estructuró en secciones, comenzando con preguntas generales y avanzando hacia aspectos específicos, es decir, su objetivo fue realizar la primera fase de la metodología CRISP-ML(Q) la cual consiste en la comprensión del negocio y los datos. Durante la entrevista, se permitió al director responder libremente y, en función de sus respuestas, se formularon preguntas adicionales para asegurar una comprensión completa del proyecto. Esta herramienta estableció las bases para desarrollar preliminarmente: la idea, plantear los objetivos, definir una pregunta de investigación y obtener herramientas fundamentales en el desarrollo del modelo QA, estos elementos formaron parte indispensable del TIC.

5.2.4.2. Crowdsourcing.

El crowdsourcing se llevó a cabo mediante un documento de acceso abierto compartido en Google Drive, al cual se permitió ingresar de forma voluntaria a los estudiantes de la carrera de Computación. Este documento facilitó que los estudiantes colaboren creando

y elaborando preguntas sobre los documentos recolectados de tareas académicas de esta carrera. La participación de los estudiantes fue una parte esencial para generar un dataset variado, asegurando que el modelo de respuesta a preguntas (QA) basado en DistilBERT pudiera entrenarse con ejemplos representativos y relevantes del contenido académico de la UNL.

5.2.4.3. Paráfrasis y el enfoque Few-shot Learning

Se implementó la técnica de paráfrasis y el Few-shot Learning como un enfoque para ampliar el conjunto de datos obtenidos y completados luego de recurrir a la técnica crowdsourcing. Few-shot Learning al implementarlo como un enfoque permitió utilizar herramientas de IA como los es ChatGPT para generar preguntas derivadas de las obtenidas en el crowdsourcing. Por ello, no se ejecutó dentro del entrenamiento del modelo, en cambio se utilizó en el proceso de creación del dataset. En este caso, se logró expandir cada una de las aportaciones originales de los estudiantes a alrededor de cinco datos adicionales por entrada, y mediante la paráfrasis se realizó la generación de variaciones y reformulaciones de las preguntas obtenidas por el Few-shot Learning, manteniendo la coherencia y relevancia del contenido. Estos datos ampliados se integraron posteriormente en el conjunto de datos SQuAD1.0, recurso base para entrenar el modelo Question Answering.

5.3. Procesamiento y Análisis de los Datos

Se implementaron varias técnicas e instrumentos para la creación y procesamiento del dataset, con el objetivo de adaptar y evaluar el modelo DistilBERT en el contexto específico de tareas académicas de la carrera de Computación de la UNL. A continuación, se detalla el proceso y la utilidad de cada técnica aplicada en el procesamiento y análisis de datos, en pocas palabras en la limpieza y preparación de los datos.

- **Recopilación de Documentos Académicos:** Para la construcción del dataset, la primera etapa fue la recopilaron tareas académicas en formato PDF. Para los documentos no disponibles en formato PDF, se utilizaron herramientas gratuitas como iLovePDF para realizar las conversiones necesarias, asegurando uniformidad y sobre todo la facilidad de procesamiento en las siguientes etapas.
- **Aplicación de la Técnica Crowdsourcing:** La técnica de crowdsourcing se empleó para generar un conjunto de preguntas basadas en los documentos académicos recopilados. Los participantes (estudiantes) generaron preguntas relevantes sobre el contenido de las tareas académicas, las cuales se verificaron y ajustaron para asegurar su calidad, intentando mejorar la calidad de los datos para asegurar la creación de un modelo con mejores resultados (véase en la sección 4.2.10.4.1 del marco teórico).

- **Expansión del Dataset con Few-shot Learning y Paráfrasis:** A partir de las preguntas obtenidas mediante crowdsourcing, se generaron aproximadamente 10 preguntas adicionales por cada entrada original, manteniendo la coherencia y relevancia del contenido, en esta actividad se hizo uso de IA generativas como lo es Claude[41] y CHATGPT[42] para la búsqueda de posibles alternativas de generar el valor de la variable “question” dentro del dataset. Este proceso aumentó significativamente el tamaño del conjunto de datos
- **Construcción del dataset en formato SQuAD1.0:** Se procedió a estructurar las preguntas ampliadas en el formato SQuAD1.0, completando los atributos de entrada mencionados en la fase 2 de la metodología:
  id: Un identificador único alfanumérico asignado secuencialmente.
  title: Conjuntos de palabras representativas del contexto de la materia.
  context: Frases o párrafos del documento donde se encuentra la respuesta, donde este debe contener en promedio un valor de aproximadamente 300 caracteres.
  question: La pregunta generada o ampliada.
  answers: Un diccionario con dos valores: el texto de la respuesta ("text") y el índice de inicio de la respuesta dentro del contexto ("answer_start"). Este último se determinó utilizando herramientas como "contadordepalabras" y validaciones adicionales con código Python.
- **Partición del Dataset:** El dataset se dividió en tres conjuntos: entrenamiento (train), validación (validation) y prueba (test), donde para asegurar aleatoriedad y buena distribución se hace uso de una semilla de 42, valor general aplicado durante todo el proceso de segmentación de los datos

Durante la fase de evaluación, se cargó el conjunto de datos *test* y el modelo QA ajustado se utilizó para generar respuestas a las preguntas del mismo dataset. Se empleó la métrica ROUGE, que incluye submétricas como precisión, recall y F-measure, para comparar las respuestas generadas por el modelo con las respuestas de referencia. Los resultados de esta evaluación proporcionaron una medida cuantitativa del rendimiento del modelo, permitiendo ajustar y mejorar el proceso de entrenamiento y optimización. Obteniendo así una visión general sobre la necesidad de enfocar atención plenamente en la construcción del dataset para asegurar el correcto funcionamiento en las siguientes fases.

[41] Claude
[42] ChatGPT

6. Resultados

6.1. Objetivo 1: Adaptar el modelo DistilBERT para responder a preguntas sobre el contenido extraído de tareas académicas en la carrera de Computación de la UNL

Los objetivos, al estar alineados con la metodología CRISP-ML(Q), se cumplen a través de tareas y actividades específicas correspondientes a cada una de sus fases. A continuación, se presentan los resultados obtenidos en cada una de ellas.

6.1.1. Comprensión de datos y negocios

Para este estudio, se utilizó la herramienta en línea Rayyan[43], que facilitó la búsqueda y revisión de literatura en el contexto de la aplicación de la técnica Fine Tuning en modelos Question Answering, permitiendo monitorear las fuentes de información necesarias para obtener los mejores resultados. Mediante la lectura y análisis de los documentos obtenidos además de la entrevista realizada al director del proyecto de vinculación, se llevó a cabo la tarea de comprensión de datos y negocios. En la **Tabla 5** se resumen las preguntas y las respuestas más sobresalientes generadas durante la entrevista.

Tabla 5. *Preguntas clave para determinar la necesidad de la creación de un modelo QA*

Preguntas clave de la entrevista al director del Proyecto "Inclusión lectora de estudiantes con discapacidad visual mediante innovación tecnológica"	
Pregunta	**Resumen de la respuesta**
¿Cuáles son los módulos o componentes principales en el proyecto?	Desarrollar un sistema de búsqueda de respuestas a preguntas (QA) para facilitar el acceso a la información contenida en tareas académicas de la carrera de Computación.
¿Qué tipo de información se desea procesar?	Documentos en formato PDF y texto plano correspondientes a trabajos, tareas e indicaciones realizadas por estudiantes en diferentes asignaturas de la carrera.
¿Recomienda usar técnicas como el Fine-Tuning o el Transfer Learning con modelos existentes, o crear un modelo desde cero?	Se sugiere utilizar técnicas de Procesamiento del Lenguaje Natural (PLN) y aprendizaje profundo, usando técnicas como el Fine-Tuning en DistilBERT.

Con base al análisis de la información obtenida durante el trascurso de esta fase (vea sección Trabajos Relacionados) y la entrevista realizada se confirmó la necesidad de desarrollar un modelo de búsqueda de respuestas utilizando la técnica de Fine-Tuning sobre el modelo QA DistilBERT, además al entender las necesidades principales del proyecto se pudo establecer preliminarmente un objetivo general y dos específicos que sirvieron como base para este TIC. En la **Tabla 6** se muestran un objetivo general y dos específicos preliminares.

[43] Herramienta de IA rayyan

Tabla 6. *Objetivos preliminares obtenidos en base a la fase de comprensión de los datos y el negocio*

Objetivos del proyecto		
General		Obtener el valor de precisión que se obtiene al ajustar un modelo QA implementando la técnica Fine-Tuning en DistilBERT para dar respuesta a preguntas sobre el contenido extraído de tareas académicas de la carrera de Computación en la UNL.
Específicos	Objetivo 1	Adaptar un modelo QA para responder a preguntas en la carrera de Computación de la UNL.
	Objetivo 2	Evaluar el rendimiento del modelo QA en cuanto a la precisión.

Para acceder a la entrevista completa ingrese al YouTube[44] o ingrese al repositorio de acceso abierto drive[45].

6.1.2. Ingeniería de datos (preparación de datos)

En esta fase se obtuvo la fuente de datos, es decir la fuente principal para el proyecto, ya que los datos es una parte muy importante en el ajuste de un modelo de ML. Aquí se realizó 3 actividades: recolección de los datos, construcción de un dataset en formato SQuAD1.0 y la división del dataset.

6.1.2.1. Tarea: Recopilar documentos de tareas académicas.

En esta etapa se recolectaron un total de 30 documentos[46] de tareas académicas correspondientes a 6 materias de la carrera de Computación de la UNL. Estos documentos en formato PDF contienen texto plano[47] en su interior, es decir, no contienen estructuras complejas como tablas o figuras, con la finalidad de facilitar su procesamiento posterior, además son tareas e indicaciones realizados durante el periodo 2021-2024, las asignaturas que sirvieron como muestra son: Ética Profesional (Ética P.), Software Quality (Software Q.), Human Perception in Computer Vision (Hpicv), Machine Learning(Machine L.), Arquitectura de Ordenadores (Arquitectura de O.) y Data Mining(Data M.). En la **Figura 12** se muestra la organización de los documentos recolectados en una carpeta de acceso abierto de drive, además en la **Figura 13** se muestra la distribución de documentos que dieron como resultado 5 archivos en formato “pdf” por materia, para plasmar esta información se usó las herramientas draw.io y Microsoft Excel respectivamente.

[44] Youtube
[45] Repositorio en GitHub: Entrevista
[46] Repositorio de documentos recolectados por materia
[47] Texto plano: información formada exclusivamente por texto

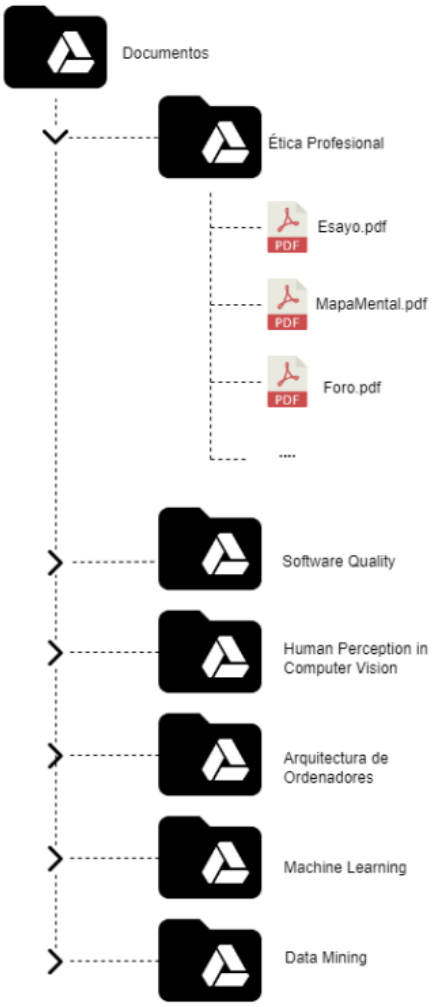


Figura 12. *Organización de los documentos recolectados*

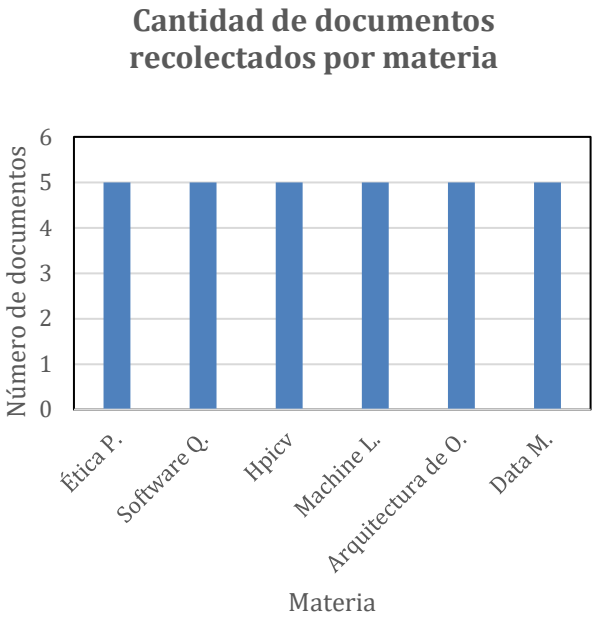


Figura 13. *Cantidad de documentos recolectados por materia*

6.1.2.2. Tarea: Construcción del dataset en formato SQUAD1.0.

Esta tarea comprende 2 actividades debido a su complejidad y esfuerzo, ya que comprende desde la obtención de preguntas hasta la construcción del conjunto de datos con el que entrenó el modelo, estas actividades son: aplicar la técnica crowdsourcing y la creación del dataset en el formato SQuAD1.0.

- **Aplicar la técnica crowdsourcing**

El crowdsourcing fue aplicado en un documento[48] colaborativo en línea, mismo que se estructuró con 4 columnas o variables: nombre(opcional), nombre del documento del que obtiene la pregunta(obligatorio), pregunta(obligatorio) y respuesta(opcional). La **Tabla 7** muestra la estructura con un ejemplo del formato usado para aplicar la técnica crowdsourcing en la recolección de preguntas sobre los documentos recolectados en la tarea anterior, donde se obtiene la recolección de aproximadamente 90 preguntas resultantes de la colaboración

[48] Repositorio en Drive con la técnica crowdsourcing

de estudiantes. Es importante destacar que la técnica crowdsourcing permitió obtener una variedad de perspectivas, ideas y enfoques al generar las preguntas, lo que enriquece la calidad del dataset.

Tabla 7. *Formato del documento compartido en el drive para realizar la técnica crowdsourcing*

Extracto y estructura de una fila del documento para el crowdsourcing			
Participante (opcional)	**Nombre del documento (obligatorio)**	**Pregunta (obligatorio)**	**Respuesta (opcional)**
Edy Jiménez	ética2	¿Cuál es el título del ensayo?	Ensayo Cuestionamientos de la Ética y Moral profesional.

- **Creación del dataset en formato SQuAD1.0**

El conjunto de datos de las tareas académicas para el modelo QA se desarrolló en formato SQuAD1.0, y se compone por las variables: id, title, context, question y answers. En esta última variable se dispone de dos valores, el texto de respuesta y el índice de inicio de la respuesta. En la realización del dataset se establecen algunos criterios de inclusión y exclusión del tipo de datos e información que aportan al dataset. En la **Tabla 8** los criterios de inclusión y exclusión de los datos usados en la construcción conjunto de datos en cuestión.

Tabla 8. *Criterios de inclusión y exclusión en la creación del dataset SQuAD1.0*

Criterios de inclusión y exclusión en el proceso de creación del dataset		
Variable	**Inclusión**	**Exclusión**
Id	• Tipo de dato alfanumérico • Tener secuencia y formato similar al id anterior	• Identificadores duplicados.
title	• Debe reflejar el contenido del contexto de manera precisa.	• Títulos ambiguos o no representativos del contenido
context	• Debe ser una oración, frase o párrafo y estar en un rango entre 30 a 1000 tokens	• No deben ser palabras unitarias o letras
question	• Tener los componentes: interrogante, tema y contexto • Debe estar relacionada con el contexto proporcionado.	• Preguntas que requieren inferencias o no están claramente presentes en el texto.
answers	• Frases específicas del contexto, precisa. • El índice de respuesta debe ser mayor a 0	• No se encuentra en el contexto, requiere inferencia, errores gramaticales
general[49]	• Idioma español • Información encontrada en los documentos • Información en formato texto plano	• Errores ortográficos significativos

[49] general: este elemento no corresponde a una variable de SQuAD1.0, en esta variable se exponen los criterios de inclusión y exclusión a nivel general del dataset

Luego de obtener los criterios de inclusión y exclusión se realizó la implementación del enfoque Few-shot Learning y la paráfrasis. En la **Figura 14** se presenta el proceso de aumento de datos, desde la herramienta CHATGPT para implementar el enfoque Few-shot Learning para generar 5 salidas de preguntas en base a una base y mediante técnica de paráfrasis se crea un dato adicional y se conserva el generado, por lo que se logra obtener 10 datos en base a un dato original. En la **Tabla 9** se presenta la estructura del dataset con valores en cada variable en base a los criterios de inclusión y exclusión mencionados en la **Tabla 8**.

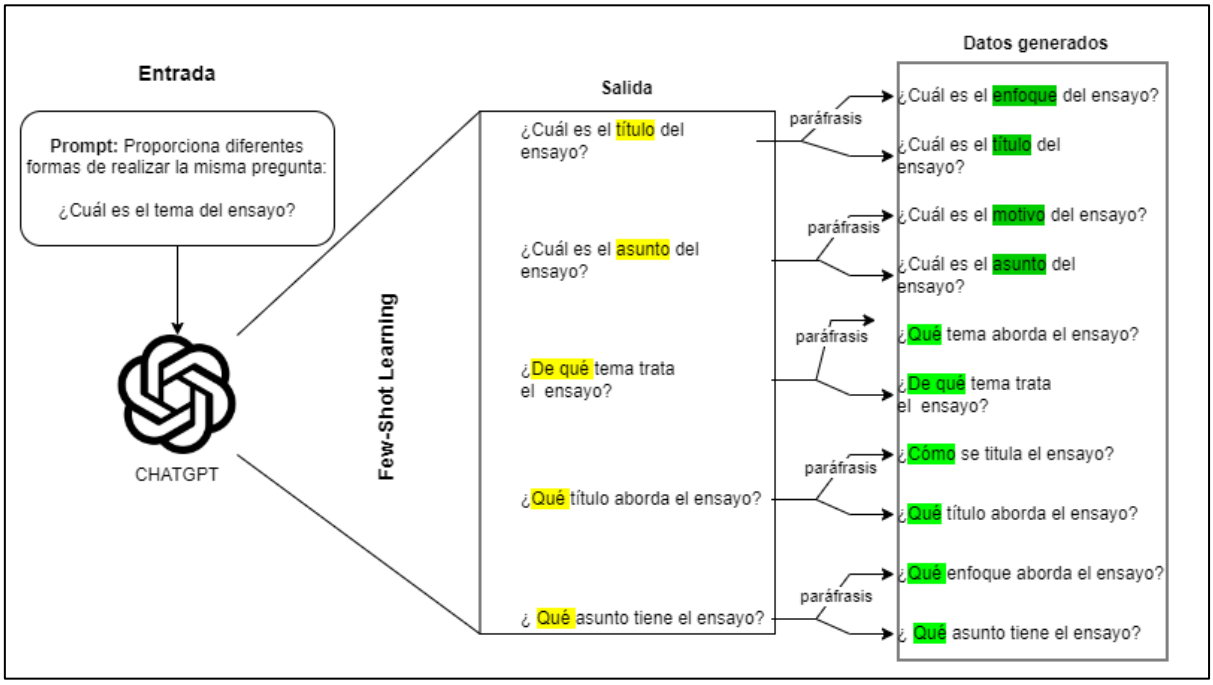


Figura 14. *Ejemplificación del enfoque Few-shot Learning y la paráfrasis en el aumento de datos*

Tabla 9. Estructura del dataset en formato SQuAD 1.0

Formato SQuAD1.0				
id	**tittle**	**context**	**question**	**answers**
5727dd9c4b86 4d1900163f140	Security11	Objetivo general: Evaluar la seguridad de una aplicación web.	¿Cuál es el objetivo general?	{'text': ['Evaluar la seguridad de una aplicación web'], 'answer_start': [18]}

En la **Figura 15** se puede visualizar información en cuanto al contenido estructurado el dataset en el repositorio[50] , exactamente en un documento Excel online, almacenando 1410 filas que contienen información sobre las variables de SQuAD1.0.

[50] dataset

id	title	context	question	answers
5727dd9c4b864d1	Human Percep	La tarea podrá ser realizará en co	¿Qué plantilla se debe utilizar?	{ 'text': ['la plantilla facilitada en Latex en Overleaf'], 'answer_
5727dd9c4b864d1	Human Percep	La tarea descrita deberá realizarla	¿A qué plataforma se pide enviar la tare	{ 'text': ['a la plataforma EVA-UNL'], 'answer_start': [113] }
5727dd9c4b864d1	Human Percep	La tarea descrita deberá realizarla	¿Hasta qué hora se debe enviar la tarea	{ 'text': ['hasta las 23:59 pm '], 'answer_start': [137] }
5727dd9c4b864d1	Human Percep	La tarea descrita deberá realizarla	¿En qué fecha se debe entregar la tarea	{ 'text': ['Lunes 13 de Noviembre de 2023.'], 'answer_start': [1
5727dd9c4b864d1	Human Percep	La tarea será presentada de forma	¿De qué forma debe ser presentada la ta	{ 'text': ['de forma individual'], 'answer_start': [25] }
5727dd9c4b864d1	Human Percep	La tarea es individual y deberá ex	¿Qué se debe anexar en la tarea?	{ 'text': ['las pantallas necesarias del resultado final'], 'answer
5727dd9c4b864d1	Human Percep	Realice el análisis y revisión de dc	¿Cuántas librerías se debe analizar y re	{ 'text': ['dos librerías'], 'answer_start': [34] }
5727dd9c4b864d1	Human Percep	No olvidar que, el tamaño del arch	¿Cuál es el peso maxímo que debe tene	{ 'text': ['es máximo 2MB'], 'answer_start': [48] }
5727dd9c4b864d1	Human Percep	El Trabajo Final de Asignatura cor	¿En qué consiste El Trabajo Final de Asi	{ 'text': ['consiste en definir de forma clara su proyecto final de
5727dd9c4b864d1	Human Percep	Realizar un ejemplo de visión por	¿Qué porcentaje de precisión se debe ol	{ 'text': ['un porcentaje de precisión superior al 90%'], 'answe
5727dd9c4b864d1	Human Percep	El Trabajo Final de Asignatura cor	¿Cuándo se deberá defender El Trabajo	{ 'text': ['al final del periodo académico.'], 'answer_start': [117

Figura 15. *Fragmento del dataset de las tareas académicas de Computación de la UNL*

- **Tarea: Particionar los datos de entrenamiento, validación y test**

El dataset obtenido a la tarea anterior se lo dividió en 3 conjuntos: train(entrenamiento), validation(validación) y test(evaluación), donde el subconjunto de validación se obtuvo de una posterior división a los datos correspondientes al conjunto de datos de entrenamiento. En la **Figura 16** se muestra la estructura de división del dataset general que contiene 1410 datos en formato SQuAD1.0.

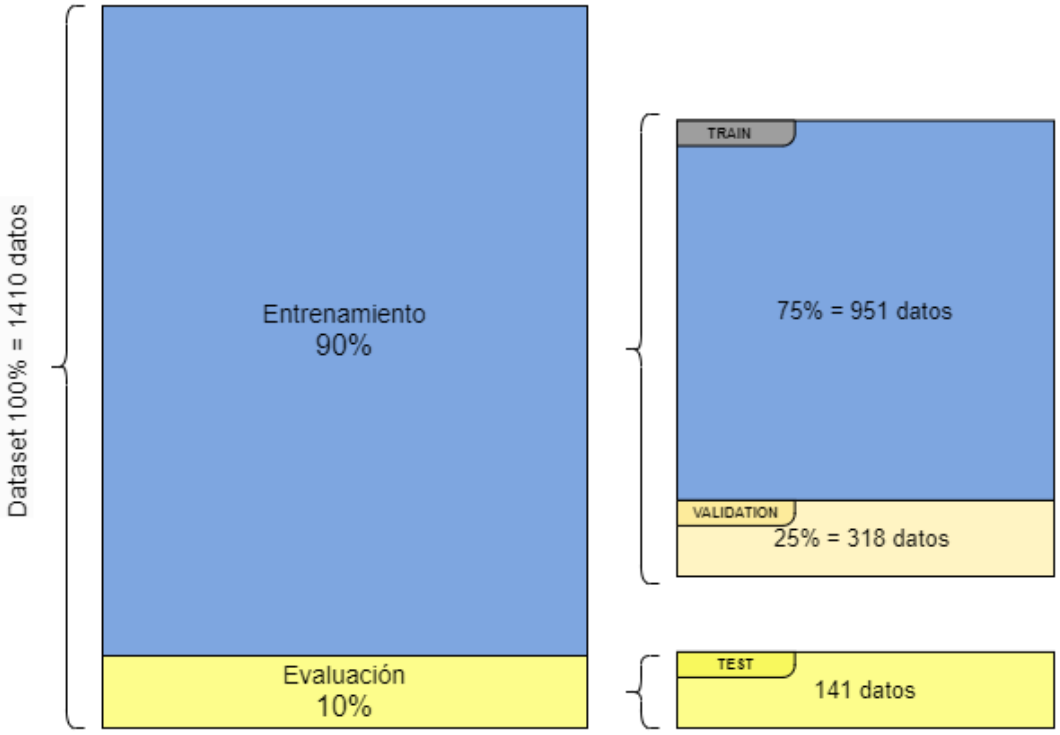


Figura 16. *División del dataset* ***datasetSQuADeS***

Mediante la partición de los datos se obtuvo la siguiente distribución:

- Conjunto de entrenamiento: 951 datos.
- Conjunto de validación: 318 datos.
- Conjunto de evaluación: 141 datos.

Estas particiones corresponden al 100% del total de los datos. En la **Figura 17** se muestra el código utilizado para la división del dataset, en el que se incluye elementos como: repositorio y ruta del conjunto de datos en formato JSON, el valor de la semilla(seed), y los porcentajes de cada uno de los dataset: train, validation y test.

```
import json
from datasets import load_dataset, Dataset, DatasetDict

# Carga el conjunto de datos desde el archivo JSON
squad_dataset = load_dataset('json', data_files='/content/drive/MyDrive/datasetSQuADeS.json')
# Como el conjunto de datos cargado está bajo la llave 'train' por defecto, lo extraemos
dataset = squad_dataset['train']
#
# Divide tu conjunto de datos en entrenamiento y prueba
train_test = dataset.train_test_split(test_size=0.1, seed=42)

# Divide el conjunto de entrenamiento en entrenamiento y validación
train_val = train_test['train'].train_test_split(test_size=0.25, seed=42)

# DatasetDict con conjuntos de entrenamiento, validación y prueba
datasets = DatasetDict({
    'train': train_val['train'],
    'validation': train_val['test'],
    'test': train_test['test']
})
```

Figura 17. *Código de división del dataset incluido los porcentajes de partición*

6.1.3. Ingeniería de modelos de aprendizaje automático

Esta fase se compone de tres actividades: entender el modelo y generar una arquitectura general del proceso del Fine-Tuning sobre DistilBERT, ajustar los hiperparámetros del modelo y realizar el entrenamiento.

6.1.3.1. Tarea: Entender el modelo QA y crear una arquitectura.

En esta tarea, se exploró y comprendió la arquitectura del modelo DistilBERT para responder preguntas (QA). En la **Figura 18** se observa la arquitectura que se desarrolló para aplicar la técnica Fine-Tuning sobre el modelo pre-entrenado DistilBERT, conservando los elementos básicos de un modelo QA.

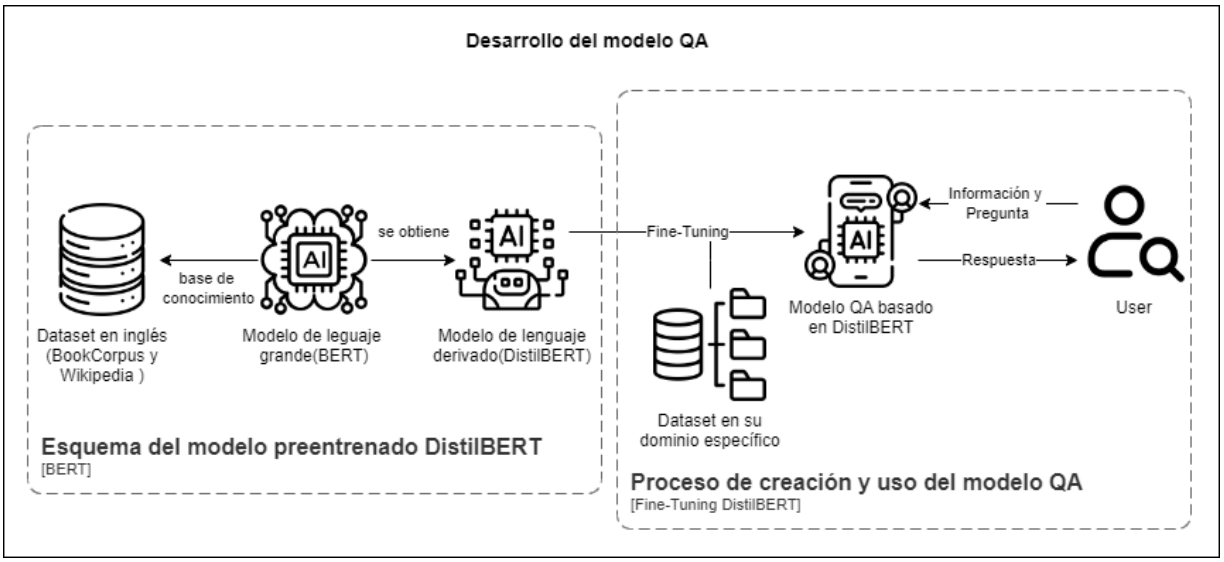


Figura 18. *Arquitectura del FineTunig para la obtención de un modelo QA*

Esta arquitectura consta de los siguientes elementos:

1. **Dataset en inglés:** corpus de información con la que fue entrenado el modelo BERT, dicha información sirve como base de conocimiento para entrenar el modelo y es obtenida de Wikipedia y BookCorpus[51].
2. **Modelo BERT:** es un modelo de lenguaje de gran tamaño entrenado previamente con la información de la base se conocimiento para adquirir conocimientos generales del lenguaje.
3. **Modelo DistilBERT:** este es el modelo pre-entrenado al que se le aplicó un ajuste, es una versión comprimida, más liviana y con un rendimiento comparado del modelo BERT, características necesarias en un entorno con pocos recursos computacionales.
4. **Fine-Tuning de DistilBERT**: técnica que se le aplicó al modelo DistilBERT utilizando un conjunto de datos específico del dominio de la carrera de computación (dataset en formato SQuAD1.0 con 1410 datos), el cual mediante el ajuste de hiperparámetros y un arduo proceso de entrenamiento se obtiene como resultado el modelo QA.
5. **Entrada de usuario y respuesta**: La arquitectura finaliza con un componente que representa la interacción entre usuario y el modelo ajustado de DistilBERT, donde se proporciona una respuesta correspondiente a la pregunta proporcionada y su respectivo contexto.

Esta arquitectura permitió aprovechar el conocimiento general adquirido por el modelo BERT en idioma inglés, y posteriormente, ajustarlo a un conjunto de datos específico del dominio de interés mediante el proceso de Fine-Tuning.

[51] Bookcorpus: un conjunto de datos que consta del texto de alrededor de 7.000 libros autoeditados disponible en HugginFace

6.1.3.2. Tarea: Ajustar los hiperparámetros del modelo DistilBERT.

Al ajustar los hiperparámetros de DistilBERT para optimizar el rendimiento en la tarea de búsqueda de respuestas (QA) en cuatro modelos diferentes se realizó una configuración para cada uno de ellos. La **Tabla 10** resume los valores ajustados correspondientes a los hiperparámetros más importantes de los modelos, mismos que se establecen dentro del código (**véase Anexo *4***) de cada uno de los cuatro experimentos. Esto se ajustó en la fase anterior a la ejecución del entrenamiento.

Tabla 10. *Valores ajustados para los 4 modelos*

Valores de los hiperparámetros en los 4 modelos QA					
		Modelo			
		modelo1	**modelo2**	**modelo3**	**modelo4**
Hiperparámetro	**Tasa de aprendizaje**	$1x10^{-5}$	$1x10^{-4}$	$1x10^{-3}$	$1x10^{-5}$
	Épocas	51	46	41	46
	Tamaño del lote	32	8	16	8
	Semilla	42	42	42	42
	dataset	datasetSQuADeS[52]			
	Optimizador	Adam			

Nota: modelo1= distilbert-base-uncased-QA1-finetuned-squad-es[53], modelo2= distilbert-base-uncased-QA2-finetuned-squad-es[54], modelo3= distilbert-base-uncased-QA3-finetuned-squad-es[55] y modelo4= distilbert-base-uncased-QA4-finetuned-squad-es[56]. Modelos desplegados en HugginFace

6.1.3.3. Tarea: Entrenar los modelos con el dataset creado.

En la **Figura 19** se muestra el código utilizado para entrenar el modelo QA y su posterior despliegue en la plataforma Hugging Face, guardando datos importantes como los logits y valores de entrenamiento en cada época. Usando como medio de almacenamiento la memoria temporal de Google Colab, por lo que se realizó una copia en el almacenamiento permanente de la cuenta de Google utilizada en esta fase.

[52] Dataset en GitHub
[53] Modelo1: *distilbert-base-uncased-QA1-finetuned-squad-es*
[54] Modelo 2: *distilbert-base-uncased-QA2-finetuned-squad-es*
[55] Modelo 3: *distilbert-base-uncased-QA1-finetuned-squad-es*
[56] Modelo 4: *distilbert-base-uncased-QA4-finetuned-squad-es*

```
from transformers.keras_callbacks import PushToHubCallback
from tensorflow.keras.callbacks import TensorBoard

push_to_hub_callback = PushToHubCallback(
    output_dir="./qa_model_save",
    tokenizer=tokenizer,
    hub_model_id=push_to_hub_model_id,
)

tensorboard_callback = TensorBoard(log_dir="./qa_model_save/logs")

callbacks = [tensorboard_callback, push_to_hub_callback]

model.fit(
    train_set,
    validation_data=validation_set,
    epochs=num_train_epochs,
    callbacks=callbacks,
)
```

Figura 19. *Código utilizado para el entrenamiento del modelo QA*

Durante el entrenamiento se registró los datos en todos los valores estadísticos en cada una de las épocas correspondientes a cada entrenamiento (ver **Anexo *5***). En la **Tabla 11** se muestra un resumen de los valores obtenidos en la última época de cada uno de los modelos que se entrenó, además se incluye los valores de extracción y el F-measure obtenido usando la herramienta de Python "*metrics"* de la biblioteca *datasets* luego de haber concluido la fase de entrenamiento.

Tabla 11. *Resumen de los datos estadísticos en los 4 experimentos*

Valores durante el entrenamiento en los 4 modelos QA					
		Modelo			
		modelo1	**modelo2**	**modelo3**	**modelo4**
Componentes	**Pérdida durante el entrenamiento**	0,2131	**0,0138**	5,9545	0,0931
	Precisión de los logits finales en el entrenamiento	0,9224	**0,9947**	0,0032	0,9559
	Precisión de los logits de inicio en el entrenamiento	0,9310	**1,0000**	0,0000	0,9685
	Pérdida durante la validación	**1,0588**	1,7511	5,9506	1,2632
	Precisión de los logits finales en la validación	**0,8088**	0,7931	0,0000	**0,8088**
	Precisión de los logits de inicio en el entrenamiento	**0,8150**	0,7994	0,0063	0,8088
	Época	51	46	**41**	46
	Nivel de extracción	**54,0881**	50,9434	0,0000	52,2013
	F1 score	**73,6556**	72,1887	11,6923	72,6390
	Tiempo aproximado(min)	57,10	57,13	**45,82**	54,72

Nota: Los valores marcados en negrita son los valores máximos obtenidos por un modelo en cada componente evaluado.

Luego de haber realizado la comparación entre los valores de los componentes obtenidos por los modelos, se seleccionó el mejor modelo en base a las estadísticas alcanzadas durante la etapa de entrenamiento, sobre todo en términos de extracción y F-measure de la validación, por lo que se puede decir con certeza que de los 4 modelos entrenados con diferentes valores en los hiperparámetros el modelo "modelo1: *distilbert-base-uncased-QA1-finetuned-squad-es",* fue el que se destacó siendo el que mayores valores obtuvo en 5 de los 10 componentes analizados con un valor de extracción 54,09 aproximadamente y un F-measure de 73,66. En la **Tabla 12** se presentan los valores en los hiperparámetros en el ajuste del modelo seleccionado y a su vez en la **Tabla 13** se muestran los datos de las tres primeras y tres últimas épocas luego del entrenamiento del modelo seleccionado.

Tabla 12. *Valores del ajuste con el modelo seleccionado*

Hiperparámetros modificados en el modelo QA DistilBERT	
Hiperparámetros	**Valor modificado**
Learning Rate (Tasa de aprendizaje)	$1x10^{-5}$
Epochs (Épocas)	51
Batch Size (Tamaño del lote)	32
Seed (Semilla)	42
dataset (conjunto de datos)	datasetSQuADeS.json[57]
Optimizador	Adam
Valores adicionales	
Weight Decay (Disminución del peso)	0,01
max_length (longitud máxima)	384
doc_stride(solapamiento)	128

1. **Learning Rate:** Se estableció en $1x10^{-5}$, un valor estándar en la técnica Fine-Tuning de modelos pre-entrenados QA como es el caso DistilBERT.
2. **Epochs:** Se entrenó el modelo durante 51 épocas, lo que permitió alcanzar una convergencia adecuada sin sobreajuste.
3. **Weight Decay:** Se configuró en 0.01 para regularizar el peso de las conexiones neuronales y evitar el sobreajuste.
4. **Batch Size:** Se utilizó un tamaño de lote de 32, un equilibrio entre la eficiencia computacional y la precisión del modelo.
5. **Seed:** Se fijó la semilla en 42 para garantizar la reproducibilidad de los resultados y en la subdivisión de los datos en train, validation y test.
6. **max_length:** Se estableció en 384, una longitud adecuada para manejar oraciones y párrafos de tamaño mediano.

[57] Dataset en github

7. **doc_stride:** Se configuró en 128, permitiendo que el modelo capture información contextual relevante a lo largo de la información que proporciona el usuario.
8. **dataset:** Se utilizó el conjunto de datos "datasetSQuADeS.json ", luego de realizar la conversión de un archivo en formato “csv” a JSON, con ayuda de la herramienta de Python “json”.

Tabla 13. *Resumen de los datos durante el entrenamiento del modelo QA ajustado*

Entrenamiento del modelo QA						
Train Loss	**Train End Logits Accuracy**	**Train Start Logits Accuracy**	**Validation Loss**	**Validation End Logits Accuracy**	**Validation Start Logits Accuracy**	**Epoch**
5,1787	0,0571	0,0496	4,3181	0,1724	0,1818	0
3,6307	0,25	0,1810	2,8944	0,3793	0,2476	1
2,5094	0,3998	0,3147	2,1436	0,4514	0,3793	2
.....	.....	.....	.....	.....	.....	.
0,2184	0,9138	0,9397	1,0568	0,8119	0,8088	48
0,2087	0,9106	0,9375	1,0588	0,8088	0,8150	49
0,2131	0,9224	0,9310	1,0588	0,8088	0,8150	50

A continuación, se detalla cada uno de los componentes que visualizan en la **Tabla 13,** estos elementos son aquellos que proporcionó el modelo QA luego de su entrenamiento, en el que se resalta el 0,9310 de precisión en cuanto a los logits de inicio de respuesta.

- **Precisión de los logits finales por época (epoch_end_logits_accuracy)**: Este elemento compara los valores obtenidos en cuanto a la precisión del dataset de entrenamiento y de validación, donde se destaca que el valor máximo de entrenamiento y el más estable es obtenido en la época 51 con un valor de 0,9224, es decir, un 92,24% de precisión para detectar los logits finales de las respuestas en el modelo QA, mientras que en el dataset de validación en la época 51 le corresponde un 0, 8088 lo que representa un 80,88% de precisión, dando como resultado una media de 86,56% de este elemento durante la etapa completa de entrenamiento. En la **Tabla 14** se muestra el resumen de los valores obtenidos respecto a las primeras y últimas etapas de entrenamiento, caso similar ocurre en la **Figura 20** donde se puede observar la tasa de aprendizaje respecto a la precisión de la obtención de los logits finales de respuesta del modelo QA durante las 50 épocas de entrenamiento.

Tabla 14. *Resumen de valores de precisión respecto a los logits finales en entrenamiento*

Precisión de los logits finales durante el entrenamiento (Epoch_end_logits_accuracy)			
Dataset	**Valor mínimo**	**Valor máximo**	**Valor Final**
Entrenamiento(train)	0,0571	0,9194	0,9224
Validación(validation)	0,1724	0,8095	0,8088
Promedio	0,1148	0,8645	**0,8656**

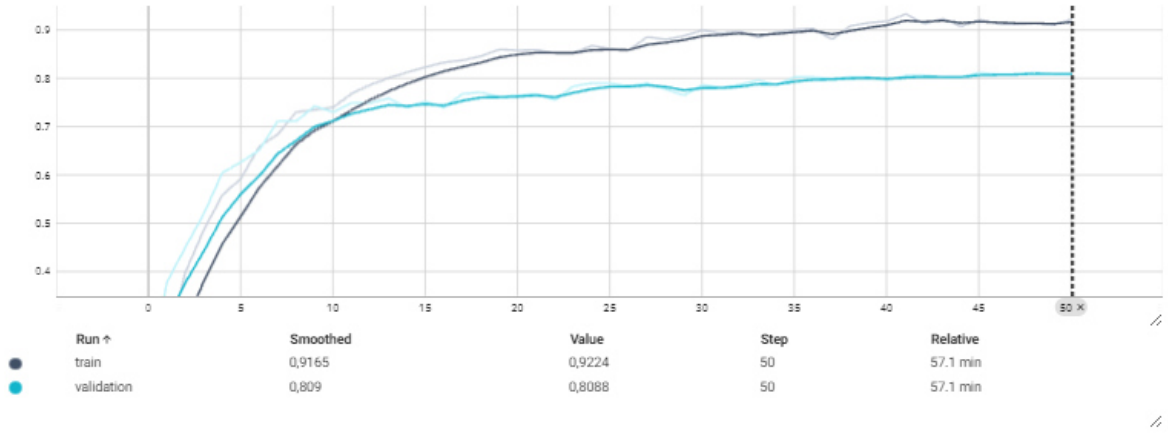


Figura 20. *Precisión de los logits finales del modelo durante el entrenamiento en 51 épocas*

- **Pérdida por época(epoch_loss):** Se obtuvo la pérdida calculada al final de cada época durante el proceso de entrenamiento, donde se destaca una pérdida de 0,2131 en el dataset de entrenamiento en la época 51, mientras que en el dataset de validación en la misma época le corresponde un 1,0588 dando como resultado una media ponderada de 0,3503. En la **Tabla *15*** se muestra el resumen de los valores mínimos, máximos alcanzados y el valor promedio de estos, caso similar ocurre en la **Figura *21*** donde se puede observar la tasa de pérdida obtenida durante el entrenamiento en la sección de train y validation.

Tabla 15. *Resumen de los valores de pérdida del modelo*

Valores de pérdida (Epoch_loss)			
Dataset	**Valor mínimo**	**Valor máximo**	**Valor Final**
Entrenamiento(train)	0,2146	5,1787	0,2131
Validación(validation)	0,9884	4,3181	1,0588
Promedio	0,6015	4,7484	**0,6360**

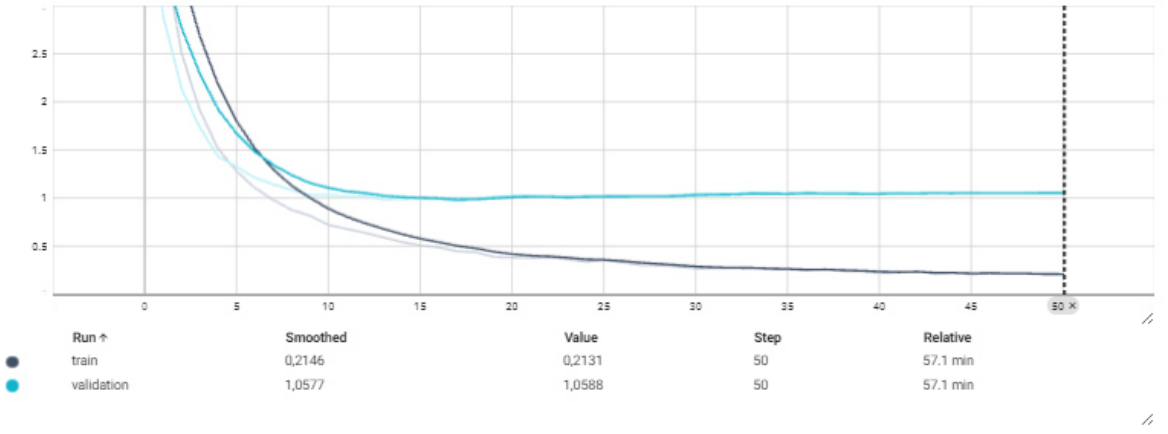


Figura 21. *Pérdida del modelo durante el entrenamiento durante 51 épocas*

- **Precisión de los logits iniciales por época (epoch_start_logits_accuracy):** En cuanto a la precisión de los logits iniciales de las respuestas del modelo QA por época durante el entrenamiento, se observan resultados notables. El conjunto de datos de entrenamiento ("train") alcanzó una precisión del 93,10% (0,9310), mientras que el conjunto de validación ("validation") registró un 81,50% (0,8150). Estos valores resultan en una media ponderada del 87,30% para este elemento durante la fase de entrenamiento. En la **Tabla 16** se muestra el resumen de los valores obtenidos respecto al valor mínimo, valor máximo y el valor que se guarda en el modelo, caso similar ocurre en la **Figura 22** donde se puede observar la tasa de aprendizaje respecto a la precisión de la obtención de los logits de inicio de respuesta del modelo QA durante las 51 épocas de entrenamiento.

Tabla 16. *Resumen de los valores de precisión de los logits de inicio durante el entrenamiento*

Precisión de los logits de inicio (Epoch_start_logits_accuracy)			
Dataset	**Valor mínimo**	**Valor máximo**	**Valor Final**
Entrenamiento(train)	0,0496	0,9336	0,9310
Validación(validation)	0,1818	0,8127	0,8150
Promedio	0,1157	0,8732	0,8730

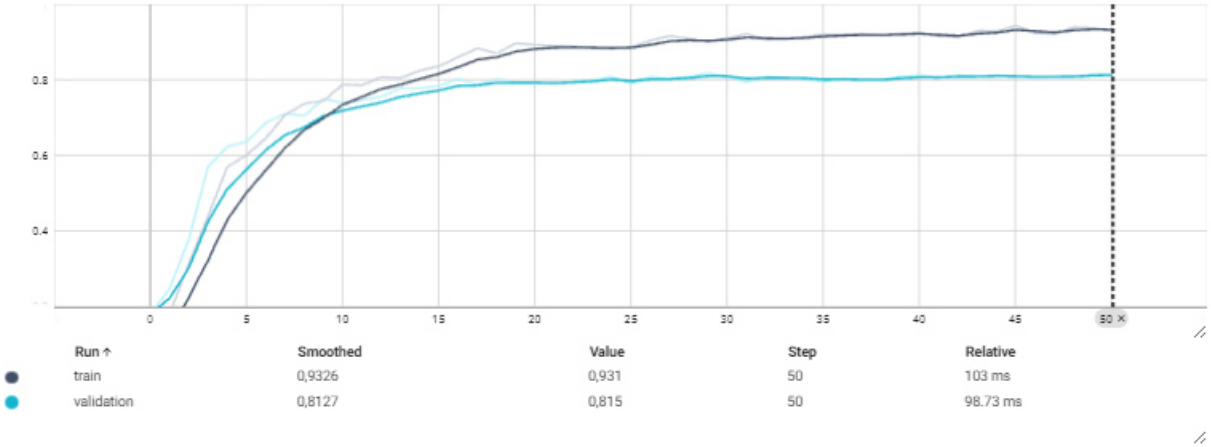


Figura 22. *Precisión de los logits de inicio durante el entrenamiento de los datos "train" y "validation" a lo largo de 51 épocas*

- **Precisión de los logits finales en la evaluación respecto a las iteraciones (evaluation _end_logits_accuracy_vs_iterations):** En relación con la precisión de los logits iniciales de las respuestas del modelo QA por época durante la validación, se observan los siguientes resultados: la precisión mínima registrada fue de 0,1724 (17,24%), mientras que el valor máximo alcanzado fue de 0,8095 (80,95%). Estos datos arrojan una media ponderada de 49,10%. Cabe destacar que el valor finalmente almacenado en el modelo QA fue de 0,8088, lo que equivale a una precisión del 80,88% en la etapa de validación, es decir, en la última fase del entrenamiento. La

Figura *23* ilustra la tasa de aprendizaje en relación con la precisión en la obtención de los logits finales de respuesta del modelo QA durante la etapa de validación.

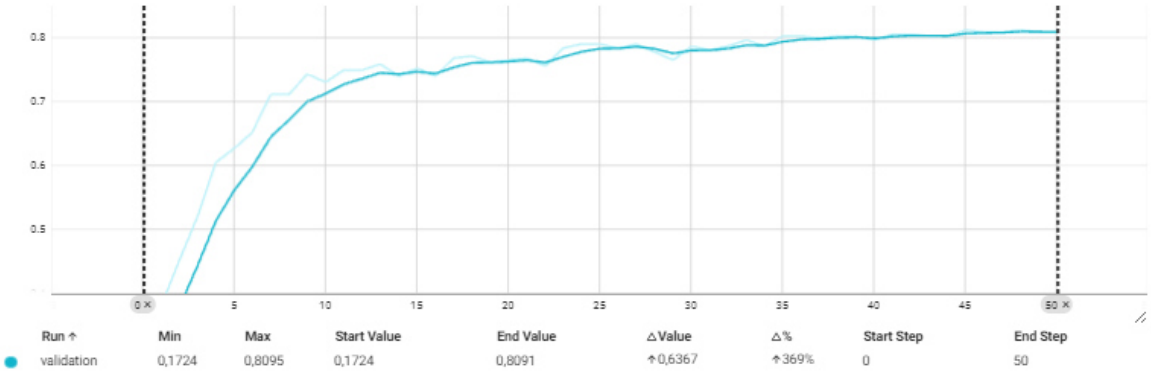


Figura 23. *Curva de aprendizaje respecto a la precisión de los logits finales en la validación*

- **Pérdida en la evaluación respecto a las iteraciones (evaluation_loss_vs_iterations):** Durante la fase de validación del entrenamiento se obtuvo al final de cada época el valor de su perdida, revelando resultados significativos. La pérdida mínima registrada fue de 0,9884, correspondiente a la época 18. En contraste, el valor máximo de pérdida fue de 4,3181, encontrado en las etapas iniciales de la validación. Estos datos arrojaron una media ponderada de 2,6533 de un valor máximo de 100. Sin embargo, es importante destacar que el valor finalmente almacenado en el modelo QA fue de 1,0588 La **Figura 24** ilustra la curva de pérdida durante la validación del modelo QA, proporcionando una representación visual de esta progresión.

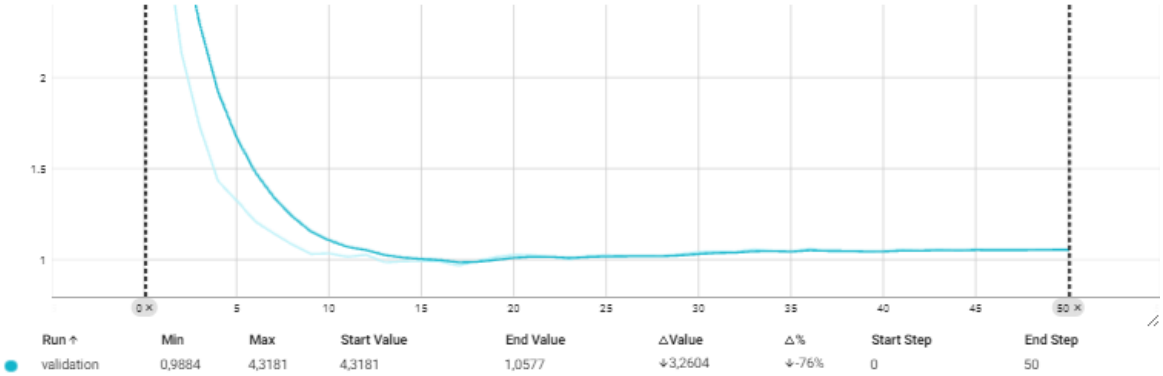


Figura 24. *Pérdida de la fase "validación" durante el entrenamiento*

- **Precisión de los logits finales en la evaluación respecto a las iteraciones (evaluation_start_logits_accuracy_vs_iterations):** En cuanto a la precisión de los logits iniciales durante la evaluación, analizada en función de las iteraciones y sobre todo enfocada en las épocas se obtuvo algunos datos destacados. La precisión mínima registrada fue de 0,1818, correspondiente a la primera época de la validación. Por otro lado, el valor máximo alcanzado fue de 0,8127, obtenido en la última época (época 51). Estos datos arrojaron una media ponderada de 0,4973. Sin

embargo, es importante destacar que el valor finalmente almacenado en el modelo QA fue de 0,8150, representando una precisión del 81,50%. La **Figura *25*** presenta la precisión durante la validación del modelo QA, ofreciendo una visualización clara de esta progresión.

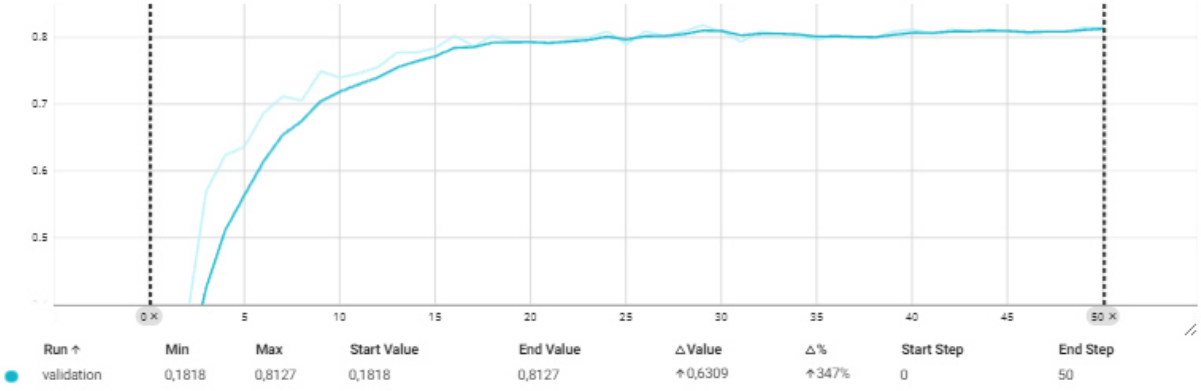


Figura 25. *Precisión de los logits iniciales respecto a las iteraciones durante la validación*

6.1.4. Modelo Question Answering

Luego del proceso de entrenamiento se obtiene que el modelo(**distilbert-base-uncased-QA1-finetuned-squad-es)**[58] fue el que brindó las mejores prestaciones**,** el cual se guardó y almacenó con sus pesos correspondientes en el archivo **"tf_model.h5**[59]**",** utilizando el conjunto de datos **"datasetSQuADeS.json**[60]**"**. En la **Tabla *17*** se presenta el resumen de los datos de entrenamiento durante la última época (51), mismos que en este caso fueron los que se guardaron como datos representativos del modelo. En la **Figura 26** se muestra el modelo desplegado en la plataforma de Hugging Face, consta de 5 secciones: *model card,* que representa los datos y la interfaz para consumir el modelo de forma directa, *files and versions,* donde se muestra los archivos necesarios para desplegar el modelo, *training metrics,* donde se carga el TensorBoard(datos obtenidos durante el entrenamiento), aquí se encuentran las gráficas estadísticas, *community,* donde se presenta un espacio de un foro para discutir aspectos referentes al modelo y la sección *settings,* donde se puede configurar algunos aspectos(idioma, dataset, entre otros) del modelo.

Tabla 17. *Valores de los elementos en el modelo QA.*

Valores de los elementos en el modelo QA	
Elemento	**Valor (4 decimales)**
Pérdida durante el entrenamiento	0,2131
Precisión de los logits finales en el entrenamiento	0,9224
Precisión de los logits de inicio en el entrenamiento	0,9310
Pérdida durante la validación	1,0588
Precisión de los logits finales en la validación	0,8088
Precisión de los logits de inicio durante la validación	0,8150
Época	51,0000

[58] Modelo en el repositorio
[59] Hugging Face: Modelo.h5
[60] Dataset en GitHub

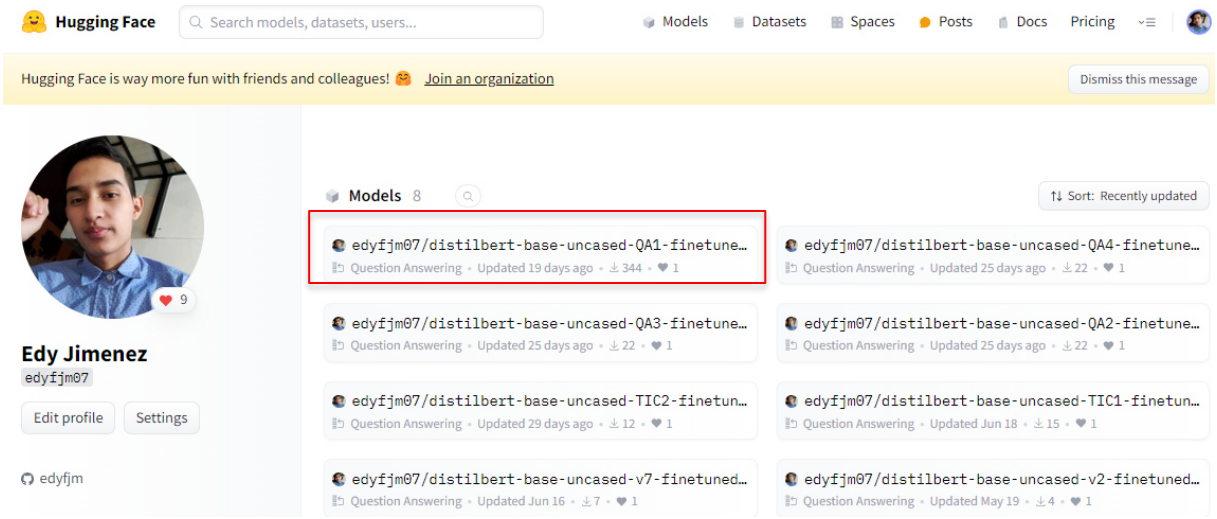


Figura 26. *Modelo QA desplegado en Hugging Face*

El modelo se encuentra desplegado en la plataforma Hugging Face, y se está disponible para hacer consultas en inglés y en español, debido a que DistilBERT fue entrenado para responder a preguntas en inglés, también se encuentra disponible para responder a preguntas en contexto del idioma español por el entrenamiento con el dataset específico en los documentos o tareas académicas de la Carrera de computación de la UNL. En la **Figura 27** se plasma el funcionamiento del modelo QA desde la plataforma Hugging Face.

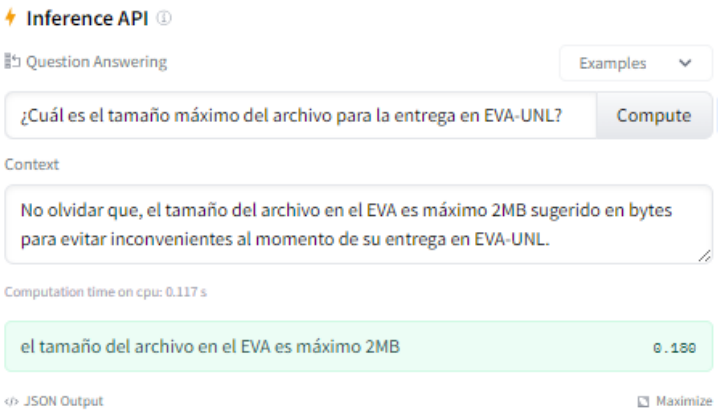


Figura 27. *Funcionamiento del Modelo QA distilbert-base-uncased-QA1-finetuned-squad-es*

Este modelo al estar desplegado en una plataforma en línea existe la posibilidad de utilizarlo mediante un pipeline[61]. En la **Figura 28** se presenta el código utilizado en Google Colab para poder utilizar el modelo QA usando el pipeline (una serie de instrucciones).

[61] Pipeline: herramienta para realizar el uso de modelos preentrenados

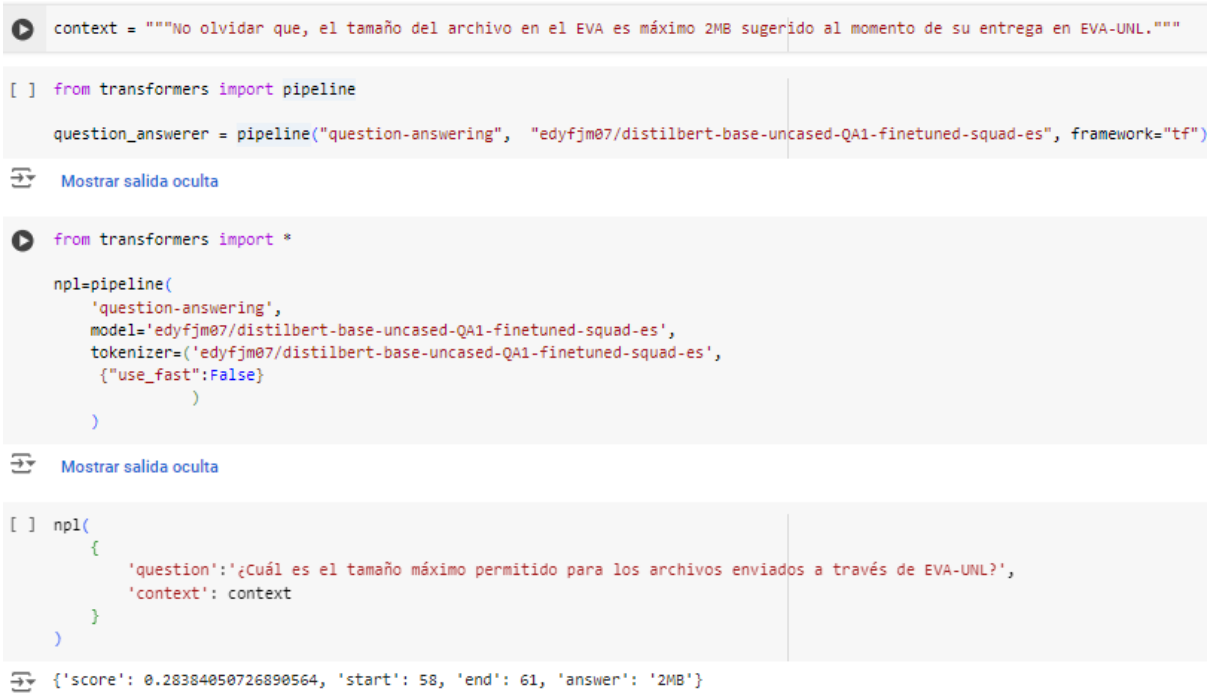


Figura 28. Código para usar el modelo QA mediante un pipeline

Instrucciones para usar el pipeline:

1. Crear y abrir un archivo con formato ipynb en Google Colab
2. Colocar una variable llamada "**context**" y ubicar la información de la cual se obtendrán las respuestas
3. Si no se redacta o pega el contenido del que quiera obtener respuestas, puede cargar el PDF, aunque primeramente se debe instalar la librería pdfminer que le permite extraer el contenido de su documento, esto puede hacerlo con el siguiente comando: **pip install pdfminer.six,** adicionalmente ubicar la ruta del documento del que desea obtener su contenido y almacenarlo con la variable "context".
4. Importar el pipeline
5. Ubicar la pregunta en la variable "question"
6. Presiona Ctrl+Enter

6.2. Objetivo 2: Evaluar el rendimiento del modelo mediante el análisis de la métrica ROUGE en la obtención de respuestas dadas por el modelo QA.

Para la presentación de los resultados, se estructuró el análisis de la métrica ROUGE en dos diferentes actividades clave. Estas incluyen la evaluación automática tanto unitaria como general, complementadas con una prueba A/B. A continuación, se detallan los resultados obtenidos en cada una de estas actividades, proporcionando una visión integral del rendimiento del modelo Question Answering para responder a preguntas sobre el contenido extraído de documentos académicos en la carrera de computación de la UNL, evaluando el dataset “test”.

6.2.1. Evaluación Automática

6.2.1.1. Unitaria

Para la evaluación de forma unitaria de los 141 datos del dataset test utilizó dos códigos. En la **Figura 29** se muestra el código para obtener la respuesta de referencia directamente del conjunto de datos. En la **Figura 30** se muestra el código para obtener las respuestas de forma individual del modelo QA.

```
[ ]  import pandas as pd
     import ast

     # Leer el archivo CSV
     df = pd.read_csv('/content/drive/MyDrive/RougeTest.csv')

     # Inicializar una lista vacía para almacenar las respuestas
     references = []

     # Iterar sobre cada fila del DataFrame
     for index, row in df.iterrows():
         # Convertir la cadena de la columna 'answers' a un diccionario
         answers_dict = ast.literal_eval(row['answers'])
         # Extraer el texto de la respuesta y agregarlo a la lista
         references.append(answers_dict['text'][0])
```

Figura 29. *Código en Python para la obtención de respuestas de referencia*

```python
from transformers import pipeline
import pandas as pd

# Cargar el modelo QA
npl = pipeline(
    'question-answering',
    model='edyfjm07/distilbert-base-uncased-QA1-finetuned-squad-es',
    tokenizer=('edyfjm07/distilbert-base-uncased-QA1-finetuned-squad-es', {"use_fast": False})
)

# Leer el archivo CSV
df = pd.read_csv('/content/drive/MyDrive/RougeTest.csv')

# Inicializar una lista vacía para almacenar las respuestas
respuestas = []

# Iterar sobre cada fila del DataFrame
for index, row in df.iterrows():
    # Obtener el contexto y la pregunta de la fila actual
    contexto = row['context']
    pregunta = row['question']

    # Obtener la respuesta del modelo para el contexto y la pregunta
    respuesta = npl({'question': pregunta, 'context': contexto})

    # Agregar la respuesta a la lista
    respuestas.append(respuesta['answer'])

# Ahora la variable 'respuestas' contiene todas las respuestas generadas por el modelo
print(respuestas)
```

Figura 30. *Código para la obtención de respuestas del modelo*

Mediante la biblioteca *rouge_score* y la comparación de forma individual de las dos respuestas: respuesta de referencia y respuesta del modelo, se obtienen valores de ROUGE en cada fila (dato del dataset *test*). En la **Tabla 18** se muestra los valores obtenidos al implementar la métrica ROUGE mediante la biblioteca *rouge_score sobre los datos de la primera fila del dataset*, esto en base a cada implementación de la métrica ROUGE de forma unitaria en los 141 datos (**véase Anexo *8***)

Tabla 18. *Valores de la métrica ROUGE implementada de forma automática.*

Obtención de los valores de la métrica ROUGE			
Pregunta	**Respuesta de referencia**	**Respuesta del modelo**	**Valor de ROUGE**
¿Cómo se puede expandir aún más el mapa mental?	Agregando subramas o detalles adicionales	agregando subramas o detalles adicionales	{'rouge1': AggregateScore(low=Score(precision=1.0, recall=1.0, fmeasure=1.0), mid=Score(precision=1.0, recall=1.0, fmeasure=1.0), high=Score(precision=1.0, recall=1.0, fmeasure=1.0)), 'rouge2': AggregateScore(low=Score(precision=1.0, recall=1.0, fmeasure=1.0), mid=Score(precision=1.0, recall=1.0, fmeasure=1.0), high=Score(precision=1.0, recall=1.0, fmeasure=1.0)), 'rougeL': AggregateScore(low=Score(precision=1.0, recall=1.0, fmeasure=1.0), mid=Score(precision=1.0, recall=1.0 fmeasure=1.0), high=Score(precision=1.0,

Pregunta	Respuesta de referencia	Respuesta del modelo	Valor de ROUGE
			recall=1.0, fmeasure=1.0)), 'rougeLsum': AggregateScore(low=Score(precision=1.0, recall=1.0, fmeasure=1.0), mid=Score(precision=1.0, recall=1.0, fmeasure=1.0), high=Score(precision=1.0, recall=1.0, fmeasure=1.0))}

6.2.1.2. General

Mediante los códigos en la evaluación automática de forma unitaria, se implementó un código adicional que compara todas las respuestas de referencia y las respuestas generadas por el modelo. En la **Figura 31** se muestra la forma de calcular la métrica ROUGE de forma general para los 141 datos del dataset *test*, en la que el diccionario *predictions* almacena la variable *respuestas* (respuestas del modelo), el diccionario *references* almacena la variable *references* (respuestas de referencia) y la variable *results* almacena el resumen estadístico de la métrica antes mencionada. En **Tabla 19** se muestran los resultados de los valores obtenidos luego de ejecutar la evaluación del conjunto de datos *test.*

```
[ ]  references=references.copy()
     respuestas=respuestas.copy()
```

```
[ ]  predictions=[respuestas]
     references=[references]
     results = rouge.compute(predictions=predictions,
                             references=references)
```

Figura 31. *Código de implementación ROUGE*

Tabla 19. *Implementación de la métrica ROUGE de forma Automática*

Resumen de la implementación de la métrica ROUGE de forma automática					
		Métrica ROUGE			
		ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-LSum
Valores de las submétricas	**Precisión (Baja)**	0,7608	0,4993	0,7575	0,7650
	Recall (bajo)	0,4464	0,3752	0,4400	0,4404
	F-measure(bajo)	**0,4809**	**0,3868**	**0,4740**	**0,4769**
	Precisión (media)	0,8198	0,5835	0,8192	0,8193
	Recall (medio)	0,5127	0,4504	0,5155	0,5133
	F-measure(medio)	**0,5415**	**0,4599**	**0,5440**	**0,5410**
	Precisión (Alta)	0,8733	0,6558	0,8749	0,8694
	Recall (Alto)	0,5850	0,5245	0,5859	0,5792
	F-measure (Alto)	**0,6083**	**0,5287**	**0,6096**	**0,6020**

Nota: Los valores marcados en negrita son el resultado de una media armónica entre la precisión y el recall, los términos entre paréntesis indica el nivel mínimo, medio y máximo que se obtuvo.

Los resultados obtenidos mediante las métricas ROUGE, revela que ROUGE-L obtuvo el valor más alto con un valor medio de F-measure de 0,5440, equivalente al 54,40% en términos porcentuales, alcanzando su punto máximo en 0,6096. Paradójicamente, ROUGE-L

también muestra su valor mínimo en 0,4740, lo que subraya la variabilidad en la capacidad del modelo para mantener consistencia en la generación de respuestas. ROUGE-1 se posiciona como la segunda métrica más efectiva, con un F-measure medio de 0.5415 (equivalente al 54,15%) y alcanzando un pico de 0,6083. Por otro lado, ROUGE-Lsum muestra un desempeño similar al de ROUGE-1, con un F-measure medio de 0,5410, reflejando una consistencia en la calidad de las secuencias generadas. En contraste, ROUGE-2 revela un rendimiento relativamente menor con un F-measure medio de 0,4599 (45,99%), siendo el más bajo entre las métricas evaluadas. Además, su valor mínimo de 0.3868 indica fluctuaciones significativas en la precisión de las generaciones de bigramas. En la **Figura 32** se muestra los valores pico obtenidos en la evaluación de la métrica ROUGE. Par ver gráfico de tendencias y el gráfico de radar de las métricas obtenidas vea **Anexo *9***.

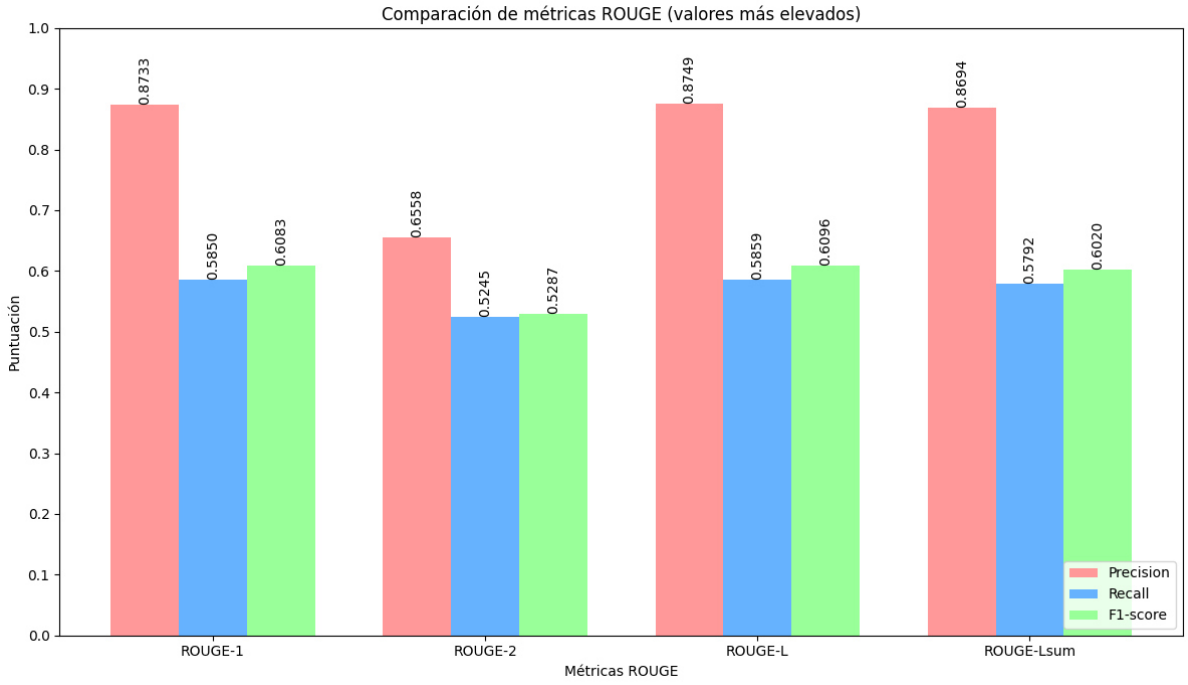


Figura 32. *Valores máximos obtenidos en la métrica ROUGE*

6.2.2. A/B testing

Los resultados del A/B testing [62]entre la métrica ROUGE (sujeto A) y las calificaciones otorgadas por un experto en PLN (sujeto B) se realizaron en una muestra ponderada de 14 datos, representando el 10% del dataset de prueba designado para la evaluación. Estos datos fueron seleccionados aleatoriamente. La **Tabla 20** presenta los valores de F-measure de la métrica ROUGE-L y las calificaciones propuestas por el experto en PLN, ambos valores en una escala de 0 a 100, donde 100 indica una respuesta completamente correcta y 0 una

[62] A/B testing en repositorio drive

respuesta incorrecta. Los valores intermedios indican respuestas parciales. Para consultar todos los datos, como el contexto, la pregunta, las respuestas generadas y de referencia, así como las puntuaciones obtenidas por ambos métodos. Al aplicar esta técnica, se obtuvo un valor promedio de 72,86 para el sujeto A y un valor de 61,43 para el sujeto B. El promedio del rendimiento en la muestra de los 14 datos seleccionados fue de 67,14. En la **Figura 33** se visualiza la tendencia de desfavorable en la calificación del profesional en PLN respecto al valor F-measure de ROUGE-L obtenido en cada una de las muestras.

Tabla 20. *Resumen de la aplicación del A/B testing*

Resumen del A/B Testing		
Nro Pregunta	**Valor F-measure de ROUGE-L/100 (Sujeto A)**	**Calificación del experto/100 (Sujeto B)**
1	100	80
2	100	90
3	100	100
4	85	100
5	100	90
6	19	10
7	20	20
8	28	0
9	70	60
10	100	100
11	100	70
12	80	60
13	42	30
14	76	50
Promedio de Unitario	72,86	61,43
Promedio Total	**67,14**	

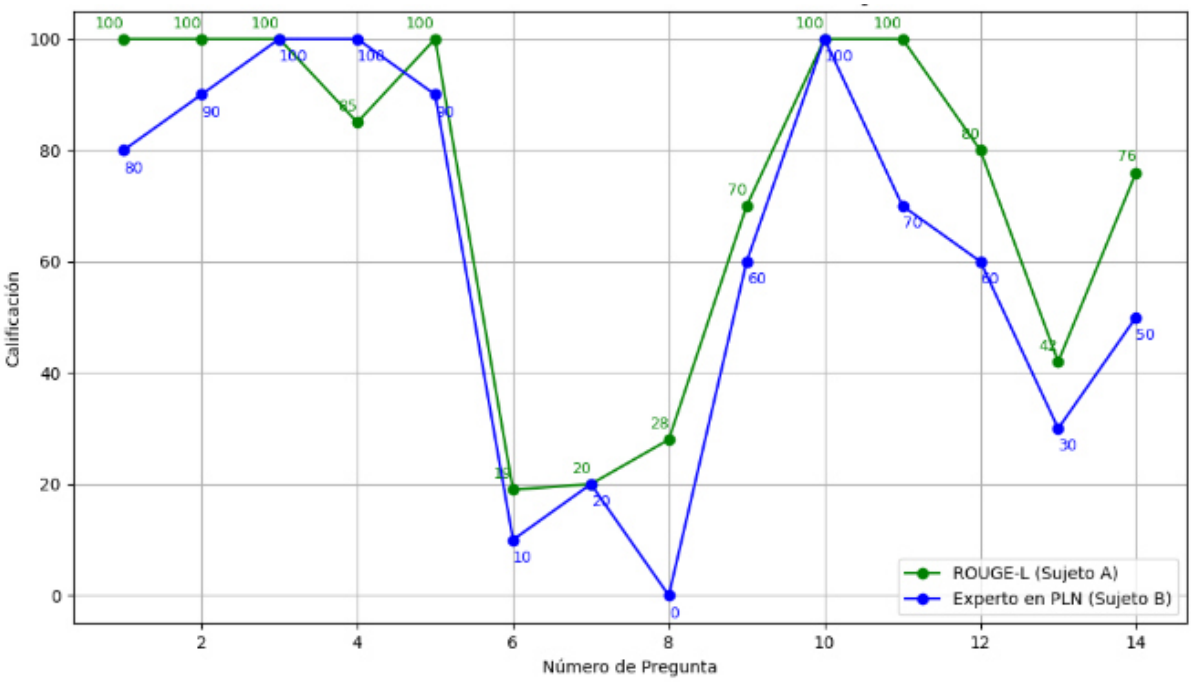


Figura 33. *Diagrama de tendencia del A/B testing*

6.2.3. Modelo aprobado y empaquetado

Como resultado final se obtiene que el primer experimento obtenido en el objetivo específico 1 fue el modelo de nombre "**ModeloQA_TIC1.ipynb**" mismo que fue desplegado en HugginFace con el nombre de "**edyfjm07/distilbert-base-uncased-QA1-finetuned-squad-es**" el cual después de la evaluación con la métrica ROUGE y mediante el A/B testing se evaluó y fue aprobado por el Ing. Oscar Cumbicus (**vea Anexo *10*).**

Para obtener esta validación, se diseñó un prototipo que permitiera ejecutar el modelo de manera local. Utilizando el framework Flask se desarrolló una interfaz gráfica de usuario, y mediante la librería Tensorflow se descargó el modelo subido en Hugging Face. En la **Figura 34** se presenta la interfaz usada para el uso del modelo QA de forma local, en la que el usuario puede redactar el contexto y preguntar sobre este, o puede cargar un documento pdf y extraer el contenido para usarlo como contexto.

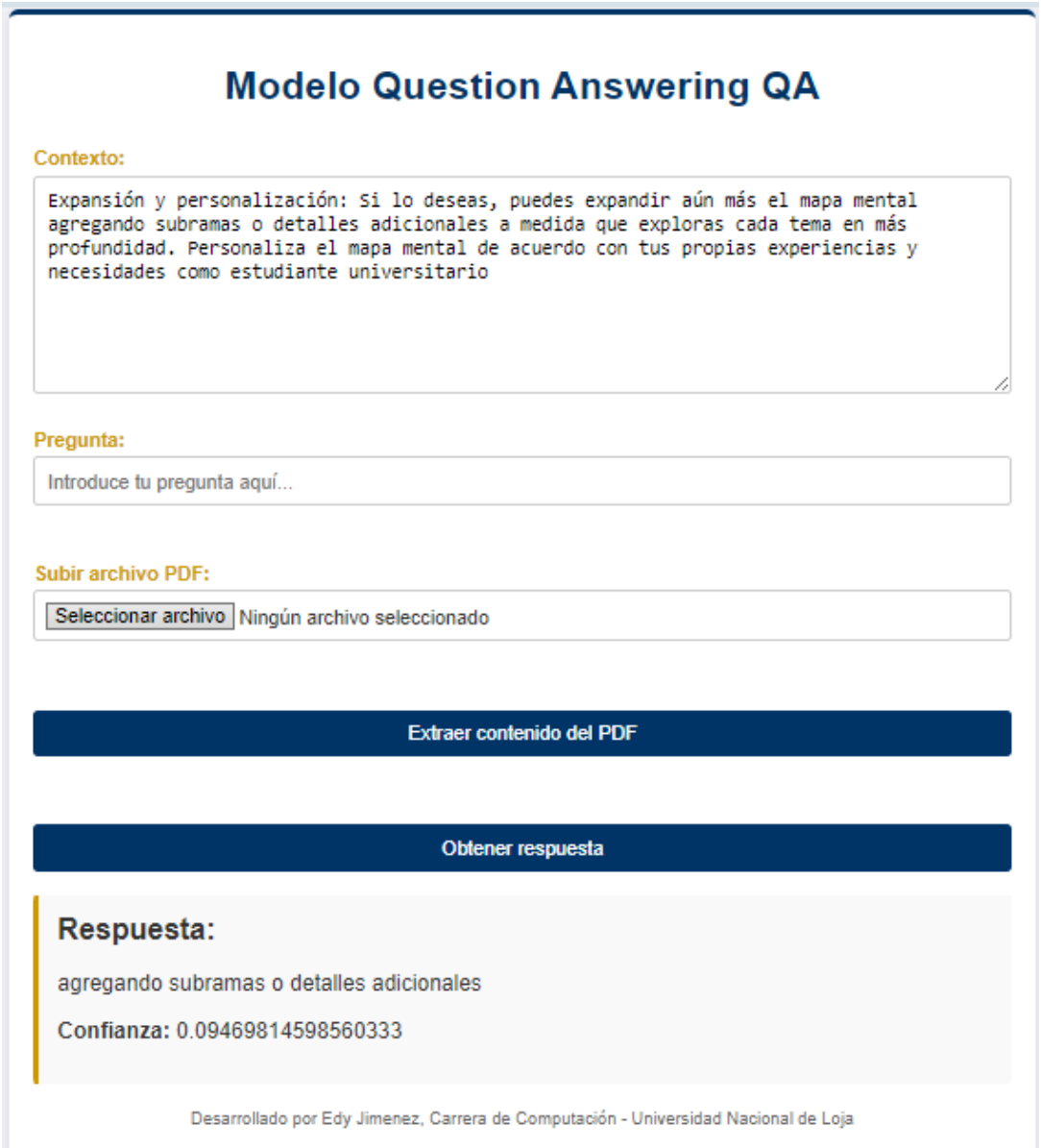


Figura 34. *Funcionamiento del modelo QA desplegado localmente*

La aplicación fue empaquetada y subida a Github[63] con el nombre **QA_local,** repositorio el cual contiene el archivo **README.md** con las instrucciones paso a paso para implementar el prototipo de manera local**.**

[63] Modelo local en github

7. Discusión

En esta sección se pretende ratificar y realizar un análisis directo entre los resultados obtenidos y la literatura tomada como base para argumentar este presente TIC, cuyo objetivo fue determinar la puntuación ROUGE que se obtiene al ajustar un modelo Question Answering implementando la técnica Fine-Tuning en DistilBERT para dar respuesta a preguntas sobre el contenido extraído de tareas académicas de la carrera de Computación en la UNL, además de obtener respuesta a la pregunta de investigación: ¿Qué puntuación ROUGE se obtiene al ajustar un modelo QA implementando la técnica Fine-Tuning en DistilBERT para dar respuesta a preguntas sobre el contenido extraído de tareas académicas de la carrera de Computación en la UNL?. A continuación, se analizan detalles importantes abarcados durante la ejecución de cada uno de los objetivos específicos establecidos en este TIC.

7.1. Primer objetivo: Adaptar el modelo DistilBERT para responder a preguntas sobre el contenido extraído de tareas académicas en la carrera de Computación de la UNL.

Debido a que la problemática trata de encontrar solución en la construcción de un modelo QA, se destaca la importancia de redes neuronales que sean eficientes en cuanto al procesamiento de texto, por lo que mediante investigación de bibliografía se propone como solución un modelo basado en redes Transformers debido a las ventajas que esta red proporciona en el procesamiento de lenguaje natural tomando como base los trabajos T1 y T2 [38],[40]. Al obtener esta información se analizó crear un modelo de PLN desde cero, y a partir de la entrevista con el profesional en temas de PLN el Ing. Oscar Cumbicus se decidió realizar una investigación de modelos eficientes que requieran poco procesamiento, es decir, seleccionar un modelo que sea adaptable a contextos específicos, y mediante la lectura de las ventajas de BERT expuestas en los trabajos T5 y T6 [25], [88] se eligió como modelo ideal a DistilBERT, por ser 60% más rápido que BERT y teniendo un 40% de tamaño inferior, además de su capacidad de adaptación en contextos específicos. Sin embargo, existió una discrepancia en la forma de entrenar al modelo QA ya que en los trabajos T5 y T6 se destaca la necesidad de un conjunto de datos amplio como SQuAD que tiene alrededor de 80000 datos, por lo que buscando soluciones se logra encontrar los trabajos T3 y T7 [3], [89] que argumentan que se puede crear un modelo especializado con una cantidad cercana a los 2000 datos, es por ello que se optó por crear un dataset variado en formato SQuAD1.0 con 1410 datos obtenidos directamente de trabajos académicos de la carrera de Computación de la UNL, con la consigna que tienen que ser lo más diversos para lograr resultados significativos. Teniendo en cuenta los materiales necesarios, se eligió Python para desarrollar el modelo, esto en base a que en todos los artículos presentados en la sección de trabajos

relacionados mencionan que este lenguaje de programación es ideal para crear o adaptar modelo de PLN especialmente en Question Answering.

Por otra parte, la elección de la metodología para desarrollar el modelo QA no estaba del todo claro debido a que metodologías como XP, Scrum y otras están enfocadas para crear sistemas de software, y metodologías como CRISP-DM se enfocan en minería de datos, pero leyendo el articulo T13[8] se optó por CRISP.ML(Q) un equilibrio entre calidad del proceso, buenas prácticas y sobre todo un proceso adaptable para ser implementado como marco de referencia en la construcción de un modelo de ML especialmente usando sus 4 primeras fases. Para obtener resultados ideales al ajustar el modelo DistilBERT se tuvo que elegir los hiperparámetros a modificar, para ello se siguió algunas pautas y opciones propuestas en el T9 y T10[50], [91], y a partir de este conocimiento se optó por un tamaño de lote de 32, una taza de aprendizaje de $1x10^{-5}$ y una cantidad de épocas de 51. El modelo Question Answering seleccionado en este estudio demostró un rendimiento sobresaliente en términos de precisión, alcanzando un 92.24% durante el entrenamiento y un 80.88% en la validación a lo largo de 51 épocas. Estos resultados son similares a estudios previos, como el T4 de Paredes y Figueroa en el 2022[6] que reportó una precisión máxima del 89,20% en el entrenamiento en un modelo QA similar al haber realizado un ajuste de solamente 3 hiperparámetros como lo son: (número de épocas, taza de aprendizaje y tamaño del lote) y un 94,50% alcanzado por Carrión y Cumbicus en el T8[7] mediante un ajuste de un modelo en los mismos 3 hiperparámetros, incluidos unos adicionales(tamaño de oración, entre otros) lo que indica relevancia e impacto de estos en el ajuste de este tipo de modelo QA.

7.2. Segundo objetivo: Evaluar el rendimiento del modelo mediante el análisis de la métrica ROUGE en la obtención de respuestas dadas por el modelo QA.

La evaluación mediante métricas ROUGE reveló una discrepancia entre la precisión y el recall de las respuestas, con una alta precisión aproximadamente con un 91.93% pero un bajo recall aproximadamente 40.46% y una media ponderada F-measure de 56.19%, tendencia que se mantuvo similar en las otras métricas ROUGE (ROUGE-2, ROUGE-L, ROUGE-LSum) esta simetría la fundamenta el T11[92]. La disparidad entre precisión y recall sugiere que, si bien el modelo es altamente eficaz en la identificación de términos correctos, enfrenta desafíos en la generación de respuestas completas y coherentes como se evidencia mediante la técnica A/B Testing la cual confirmó estas observaciones, revelando que las respuestas, aunque precisas en términos léxicos, a menudo carecían de coherencia y completitud base teórica fundamentada en el T12[5]. Estos hallazgos subrayan la complejidad al desarrollo de sistemas de QA que no solo sean precisos, sino también capaces de proporcionar respuestas integrales y contextualmente apropiadas, además de la importancia de elegir la técnica con la que se ajustará el modelo, los hiperparámetros modificados y la estructura del dataset como se evidencia en T3[3]. La elección de la metodología influye directamente en la obtención de resultados debido a los entregables y las consideraciones a tener en cuenta en cada una de sus fases, tal y como se evidencia en el T13 [8].

Para abordar estos desafíos y mejorar el modelo, se plantean varias estrategias. Estas incluyen el desarrollo de un dataset más diverso y de alta calidad, la incorporación de técnicas de generación y abstracción más avanzadas, y la implementación de métodos de evaluación que capturen no solo la precisión léxica, sino también aspectos como la coherencia, relevancia y completitud de las respuestas, logrando un equilibrio óptimo entre precisión, recall, F-measure y la calidad general de las respuestas, con el objetivo de crear modelos de QA que no solo identifiquen información correcta, sino que también proporcionen respuestas completas, coherentes y útiles en aplicaciones en un contexto real y específico.

8. Conclusiones

- La evaluación del modelo Question Answerig utilizando la métrica ROUGE demuestra que este tiene un rendimiento consistente en la extracción de respuestas sobre en un contexto de las tareas académicas de la carrera de computación de la UNL. Con un F-measure medio de 54,40% alcanzado un valor máximo de 60,96% en ROUGE-L, se observa un equilibrio entre precisión y recall en la extracción de unigramas, en este caso palabras. Aunque ROUGE-2, con un F-measure medio de 0,4599, indica que hay margen para mejorar en la generación de las respuestas, los altos valores en ROUGE-L y ROUGE-Lsum (0,5440 y 0,5410 respectivamente) subrayan la capacidad del modelo para mantener la coherencia en respuestas más completas y estructuradas, esto se complementa al obtener una calificación de las preguntas por un profesional del procesamiento de lenguaje natural obteniendo un promedio de aproximadamente el 61% al evaluar 14 muestras elegidas al azar. Estos resultados validan la eficacia del modelo en contextos académicos específicos.
- El proceso de Fine-Tuning del modelo DistilBERT demostró ser eficiente para adaptarse a contextos específicos en tareas académicas dentro de la carrera de Computación. Mediante la configuración de hiperparámetros críticos como la tasa de aprendizaje de $1x10^{-5}$, un batch_size de 32 y una semilla de 42, y utilizando un conjunto de datos diseñado específicamente para este dominio, en un entrenamiento que se llevó a cabo durante 51 épocas en una GPU T4, optimizado con el algoritmo Adam, el modelo alcanzó una precisión del 81,50% y una pérdida de 1,0588, indicativos de su capacidad para comprender y responder de manera precisa a preguntas complejas dentro del ámbito académico. Además, se logró un notable valor de extracción de respuestas de 54,0881 y un F-measure de 73,6556 durante el entrenamiento, subrayando la robustez y la efectividad del modelo en la generación de respuestas pertinentes.
- La evaluación mediante la métrica ROUGE subraya un valor máximo de precisión de 87,49% obtenido en ROUGE-1, 87,33 en ROUGE-2, 86,94% en ROUG-LSum y 65,58% en ROUGE-2, destacando una precisión en la obtención de palabras correctas en las respuestas brindadas, aunque en la totalidad de la respuesta se destaca un F-measure máximo de 0,6096 y un mínimo 0,5287, lo que indica una proyección de mejora en cuanto a la completitud de respuestas, ya que estos valores fluctúan en el 50%.
- El conjunto de datos elaborado para entrenar el modelo de QA, basado en el formato SQuAD1.0 y compuesto por 1410 datos, extraídos de 30 documentos académicos de seis materias diferentes de la carrera de Computación de la UNL, ofrece una representación diversa y enfocada en el contexto educativo, contribuyendo un aporte significativo para la construcción del modelo QA.

9. Recomendaciones

- Antes de invertir una cantidad significativa de recursos y tiempo en el ajuste de un modelo Question Answering, se recomienda realizar una prueba de concepto (PoC). Esto ayuda a determinar la viabilidad de la idea y a identificar posibles desafíos tempranamente para evaluar la viabilidad técnica y práctica del modelo.
- Se recomienda el uso de modelos pre-entrenados como DistilBERT en la implementación de sistemas de Question Answering (QA) debido a sus significativos beneficios en términos de sostenibilidad y eficiencia. Entrenar un modelo desde cero consume grandes recursos computacionales y económicos, mientras que los modelos pre-entrenados ya poseen un conocimiento previo para interpretar el lenguaje natural, adquirido en su entrenamiento inicial en vastos conjuntos de datos. Esto permite su ajuste para tareas específicas, reduciendo el tiempo y los costos de entrenamiento y minimizando el consumo energético.
- Trabajar con versiones compatibles de las herramientas y bibliotecas. Esto incluye la versión del framework TensorFlow o Pytorch, así como las dependencias necesarias. Configurar un entorno de ejecución, ya sea local o en la nube, que soporte GPU es esencial para acelerar el proceso de entrenamiento y evaluación del modelo Question Answering, sobre todo si el dataset usado es de gran tamaño. Optar por entornos como Google Colab puede ser útil, pero se debe considerar sus limitaciones en términos de tiempo y recursos.
- No se debe limitar la evaluación del modelo únicamente a métricas estadísticas como ROUGE o BLEU. Se recomienda incorporar métodos de evaluación adicionales, como el A/B Testing, para tener la capacidad de evaluar un modelo desde dos perspectivas tanto cualitativas como cuantitativas. Este enfoque complementario permite identificar áreas de mejora que las métricas tradicionales podrían no captar, asegurando una evaluación más robusta y precisa del modelo.
- Se recomienda utilizar el modelo QA creado en este TIC como base para trabajos futuros en procesamiento de lenguaje natural. Se invita a los próximos investigadores a tomar el desafío de ampliar el enfoque de este modelo, explorando nuevas aplicaciones y mejorando sus capacidades para contribuir al avance en el campo de los sistemas de preguntas y respuestas.

9.1. Trabajos Futuros

- Ampliar el dataset, actualmente incluye 1410 datos provenientes de 30 documentos académicos de seis materias diferentes, se podría extender para abarcar una mayor variedad disciplinas, temas y/o materias dentro de la carrera. Además, la inclusión de documentos de otros programas académicos podría proporcionar un conjunto de datos más robusto y diversificado, mejorando la capacidad del modelo para responder a una gama más amplia de contextos y preguntas.
- Aumentar la dificultad y variedad de las preguntas. El formato actual basado en SQuAD1.0 proporciona una estructura sólida, pero la integración de preguntas con diferentes niveles de complejidad y tipos de respuesta podría hacer que el modelo sea más versátil y aplicable en escenarios del mundo real. Por ejemplo, incorporar preguntas de opción múltiple y preguntas abiertas, podría desafiar más al modelo y evaluar su capacidad para manejar una diversidad mayor de tareas cognitivas.
- Entrenar el modelo con SQUAD2.0, este enfoque no solo permitiría al modelo extraer respuestas directamente de un contexto dado, sino que también le proporcionaría la capacidad de generar respuestas. Esto es útil ya que las preguntas pueden ser complejas y no siempre tener una respuesta directa en los materiales de referencia. Al incorporar este tipo de preguntas y respuestas, el modelo desarrollaría una capacidad para determinar cuándo falta información en el contexto para responder una pregunta.
- Comparar el rendimiento del modelo respeto a otras métricas además de ROUGE, ya que para que la evaluación de un modelo de PLN como el Question Answering tenga mayor impacto se debe considerar la calidad de respuestas en un ambiente real donde una evaluación cuantitativa debe ser apoyada por técnicas adicionales, una de ellas podría ser la evaluación directa con usuarios (estudiantes, docentes, entre otros) pertenecientes a la carrera de Computación.
- Desarrollar una interfaz de usuario accesible para interactuar con el modelo de QA. Esta interfaz podría incluir funcionalidades como la búsqueda avanzada, filtros por materia o tipo de pregunta, entre otras funcionalidades. También, el despliegue del modelo en una plataforma web o móvil garantizaría su accesibilidad desde cualquier lugar, permitiendo a los usuarios beneficiarse de sus capacidades de respuesta en el momento que lo necesiten. Además, es esencial integrar una sección de retroalimentación en la interfaz. Esta sección permitiría a los usuarios finales evaluar la calidad de las respuestas proporcionadas por el modelo. Utilizando técnicas de aprendizaje por refuerzo, el modelo podría aprender y mejorar continuamente basándose en las anotaciones y comentarios de los usuarios.

10. Bibliografía

[1] J. Devlin, M.-W. Chang, K. Lee, and K. T. Google, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in *Association for Computational Linguistics*, Minesota: Conference, Jun. 2019, pp. 4171–4186. Accessed: May 27, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.1810.04805

[2] OpenAI *et al.*, “GPT-4 Technical Report,” 2024. [Online]. Available: https://arxiv.org/abs/2303.08774

[3] P. Dasigi, K. Lo, I. Beltagy, A. Cohan, N. A. Smith, and M. Gardner, “A Dataset of Information-Seeking Questions and Answers Anchored in Research Papers,” in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, and Y. Zhou, Eds., Online: Association for Computational Linguistics, Jun. 2021, pp. 4599–4610. doi: 10.18653/v1/2021.naacl-main.365.

[4] A. Trischler *et al.*, “NewsQA: A Machine Comprehension Dataset,” in *Proceedings of the 2nd Workshop on Representation Learning for NLP*, P. Blunsom, A. Bordes, K. Cho, S. Cohen, C. Dyer, E. Grefenstette, K. M. Hermann, L. Rimell, J. Weston, and S. Yih, Eds., Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 191–200. doi: 10.18653/v1/W17-2623.

[5] N. Schluter, “The limits of automatic summarisation according to ROUGE,” in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, M. Lapata, P. Blunsom, and A. Koller, Eds., Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 41–45. [Online]. Available: https://aclanthology.org/E17-2007

[6] V. Paredes and R. Figueroa, “Chatbot para resolver inquietudes académicas sobre estudios de posgrado en la Carrera de Sistemas/Computación de la UNL,” Tesis, Universidad Nacional de Loja, Loja, 2023.

[7] J. Carrión, V. Serrano, and O. Cumbicus, “Agente virtual para brindar asistencia acerca del covid-19,” Thesis, Universidad Nacional de Loja, Loja, 2022.

[8] S. Studer *et al.*, “Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology,” *Mach Learn Knowl Extr*, vol. 3, no. 2, pp. 392–413, Jun. 2021, doi: 10.3390/make3020020.

[9] R. D. Moreno Padilla, “La llegada de la inteligencia artificial a la educación,” *Revista de Investigación en Tecnologías de la Información*, vol. 7, no. 14, pp. 260–270, Dec. 2019, doi: 10.36825/riti.07.14.022.

[10] J. Ricardo, M. Vázquez, J. Peñafiel, and Y. Assafiri, “Inteligencia artificial y propiedad intelectual,” *Revista Científica de la Universidad de Cienfuegos*, vol. 13, no. S3, pp. 362–368, Dec. 2021, Accessed: May 07, 2024. [Online]. Available: https://rus.ucf.edu.cu/index.php/rus/article/view/2490

[11] K. L. A. Yau *et al.*, “Augmented Intelligence: Surveys of Literature and Expert Opinion to Understand Relations between Human Intelligence and Artificial Intelligence,” *IEEE Access*, vol. 9, pp. 136744–136761, Oct. 2021, doi: 10.1109/ACCESS.2021.3115494.

[12] J. Wei *et al.*, “Machine learning in materials science,” Sep. 01, 2019, *Blackwell Publishing Ltd, Wiley*. doi: 10.1002/inf2.12028.

[13] S. V. Mahadevkar *et al.*, “A Review on Machine Learning Styles in Computer Vision - Techniques and Future Directions,” 2022, *Institute of Electrical and Electronics Engineers Inc.* doi: 10.1109/ACCESS.2022.3209825.

[14] P. Shinde and S. Shah, “A Review of Machine Learning and Deep Learning Applications,” *Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pp. 1–6, Aug. 2018, doi: 10.1109/ICCUBEA.2018.8697857.

[15] D. Kauchak, “Machine learning,” 2021, Accessed: May 07, 2024. [Online]. Available: https://books.google.com/books?hl=es&lr=&id=2nQJEAAAQBAJ&oi=fnd&pg=PR7&dq=machine+learning&ots=fI_7Q9VGnp&sig=_GOsQHmjzLLW6aFdKT-lzULkmJE

[16] W. Farsal, S. Anter, and M. Ramdani, “Deep learning: An overview,” *ACM International Conference Proceeding Series*, vol. 18, pp. 1–6, Oct. 2018, doi: 10.1145/3289402.3289538.

[17] C. Janiesch, P. Zschech, and K. Heinrich, “Machine learning and deep learning,” *Electronic Markets*, vol. 31, pp. 685–695, Mar. 2021, doi: 10.1007/s12525-021-00475-2/Published.

[18] M. Ponti and G. Paranhos, “Como funciona o Deep Learning,” *Tópicos em Gerenciamento de Dados e Información*, vol. 1, pp. 1–31, Jun. 2018, [Online]. Available: http://arxiv.org/abs/1806.07908

[19] Y. Bengio, Y. Lecun, and G. Hinton, “Deep learning for AI,” *Commun ACM*, vol. 64, no. 7, pp. 58–65, Jun. 2021, doi: 10.1145/3448250.

[20] P. M. Nadkarni, L. Ohno-Machado, and W. W. Chapman, “Natural language processing: An introduction,” Sep. 2011. doi: 10.1136/amiajnl-2011-000464.

[21] M. Kounte, P. Kumar, P. P, and H. Bajpai, “Analysis of Intelligent Machines using Deep learning and Natural Language Processing,” *IEEE*, Jul. 2020.

[22] A. Galassi, M. Lippi, and P. Torroni, “Attention in Natural Language Processing,” *IEEE Trans Neural Netw Learn Syst*, vol. 32, no. 10, pp. 4291–4308, Oct. 2021, doi: 10.1109/TNNLS.2020.3019893.

[23] A. S. Alammary, “BERT Models for Arabic Text Classification: A Systematic Review,” Jun. 01, 2022, *MDPI*. doi: 10.3390/app12115720.

[24] G. Jawahar, B. Sagot, and D. Seddah, “What does BERT learn about the structure of language?,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, A. Korhonen, D. Traum, and Ll. Márquez, Eds., Florence: Association for Computational Linguistics, Jul. 2019, pp. 3651–3657. doi: 10.18653/v1/P19-1356.

[25] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter,” *ArXiv*, vol. 2, no. 5, Jan. 2020, [Online]. Available: http://arxiv.org/abs/1910.01108

[26] W. Chen, N.-M. Henry, and A. Adoma, “COMPARATIVE ANALYSES OF BERT, ROBERTA, DISTILBERT, AND XLNET FOR TEXT-BASED EMOTION RECOGNITION,” *IEEE*, vol. 5, no. 17, pp. 978–983, Jun. 2020.

[27] J. Bai, R. Cao, W. Ma, and H. Shinnou, “Construction of Domain-Specific DistilBERT Model by Using Fine-Tuning,” in *International Conference on Technologies and Applications of Artificial Intelligence, TAAI*, Institute of Electrical and Electronics Engineers Inc., Dec. 2020, pp. 237–241. doi: 10.1109/TAAI51410.2020.00051.

[28] J. Mozafari, A. Fatemi, and M. A. Nematbakhsh, “BAS: An Answer Selection Method Using BERT Language Model,” *Journal of Computing and Security*, vol. 8, no. 2, Jul. 2021, doi: 10.22108/jcs.2021.128002.1066.

[29] H. Adel *et al.*, “Improving Crisis Events Detection Using DistilBERT with Hunger Games Search Algorithm,” *Mathematics*, vol. 10, p. 447, Jul. 2022, doi: 10.3390/math10030447.

[30] A. Thawani, S. Ghanekar, X. Zhu, and J. Pujara, “Learn Your Tokens: Word-Pooled Tokenization for Language Modeling,” in *Findings of the Association for Computational Linguistics: EMNLP 2023*, H. Bouamor, J. Pino, and K. Bali, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 9883–9893. doi: 10.18653/v1/2023.findings-emnlp.662.

[31] T. Limisiewicz, J. Balhar, and D. Mareček, “Tokenization Impacts Multilingual Language Modeling: Assessing Vocabulary Allocation and Overlap Across Languages,” in *Findings of the Association for Computational Linguistics: ACL 2023*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds., Toronto: Association for Computational Linguistics, Jul. 2023, pp. 5661–5681. doi: 10.18653/v1/2023.findings-acl.350.

[32] X. Song, A. Salcianu, Y. Song, D. Dopson, and D. Zhou, “Fast WordPiece Tokenization,” in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, M.-F. Moens, X. Huang, L. Specia, and S. W. Yih, Eds., Punta Cana: Association for Computational Linguistics, Nov. 2021, pp. 2089–2103. doi: 10.18653/v1/2021.emnlp-main.160.

[33] L. H. de Chavannes, M. G. K. Kongsbak, T. Rantzau, and L. Derczynski, “Hyperparameter Power Impact in Transformer Language Model Training,” in *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing*, N. S. Moosavi, I. Gurevych, A. Fan, T. Wolf, Y. Hou, A. Marasović, and S. Ravi, Eds., Virtual: Association for Computational Linguistics, Nov. 2021, pp. 96–118. doi: 10.18653/v1/2021.sustainlp-1.12.

[34] I. Staliūnaitė and I. Iacobacci, “Compositional and Lexical Semantics in RoBERTa, BERT and DistilBERT: A Case Study on CoQA,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds., Online: Association for Computational Linguistics, Nov. 2020, pp. 7046–7056. doi: 10.18653/v1/2020.emnlp-main.573.

[35] H. Xu, J. van Genabith, D. Xiong, and Q. Liu, “Dynamically Adjusting Transformer Batch Size by Monitoring Gradient Direction Change,” *ArXiv*, vol. abs/2005.02008, 2020, [Online]. Available: https://api.semanticscholar.org/CorpusID:218502425

[36] A. Gotmare, N. S. Keskar, C. Xiong, and R. Socher, “A Closer Look at Deep Learning Heuristics: Learning rate restarts, Warmup and Distillation,” *ArXiv*, Oct. 2018, [Online]. Available: https://arxiv.org/abs/1810.13243

[37] M. Popel and O. Bojar, “Training tips for the transformer model,” *arXiv preprint arXiv:1804.00247*, 2018.

[38] T. Shao, Y. Guo, H. Chen, and Z. Hao, “Transformer-Based Neural Network for Answer Selection in Question Answering,” *IEEE Access*, vol. 4, no. 2, pp. 26146–26156, Apr. 2019, doi: 10.1109/ACCESS.2019.2900753.

[39] J. Huertas-Tato, A. Martín, and D. Camacho, “SILT: Efficient transformer training for inter-lingual inference,” *ArXiv*, vol. 1, no. 2, May 2021, [Online]. Available: http://arxiv.org/abs/2103.09635

[40] A. Vaswani *et al.*, “Attention Is All You Need,” in *Conference on Neural Information Processing Systems*, Long Beach: arXiv, Aug. 2017, pp. 1–11.

[41] M. Palomino, “Los Grandes Modelos del Lenguaje basados en Transformers: revisión y aplicación práctica con ChatGPT,” Thesis, Universidad Pontificia, Madrid, 2023.

[42] M. Sukhareva, J. Eckle-Kohler, I. Habernal, and I. Gurevych, “Crowdsourcing a Large Dataset of Domain-Specific Context-Sensitive Semantic Verb Relations,” in

Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16), N. Calzolari, K. Choukri, T. Declerck, S. Goggi, M. Grobelnik, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, and S. Piperidis, Eds., Portorož, Slovenia: European Language Resources Association (ELRA), May 2016, pp. 2131–2137. [Online]. Available: https://aclanthology.org/L16-1338

[43] J. Welbl, N. F. Liu, and M. Gardner, "Crowdsourcing Multiple Choice Science Questions," in *Proceedings of the 3rd Workshop on Noisy User-generated Text*, L. Derczynski, W. Xu, A. Ritter, and T. Baldwin, Eds., Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 94–106. doi: 10.18653/v1/W17-4413.

[44] J. Michael, G. Stanovsky, L. He, I. Dagan, and L. Zettlemoyer, "Crowdsourcing Question-Answer Meaning Representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, M. Walker, H. Ji, and A. Stent, Eds., New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 560–568. doi: 10.18653/v1/N18-2089.

[45] J. Li and F. Fukumoto, "A Dataset of Crowdsourced Word Sequences: Collections and Answer Aggregation for Ground Truth Creation," in *Proceedings of the First Workshop on Aggregating and Analysing Crowdsourced Annotations for NLP*, S. Paun and D. Hovy, Eds., Hong Kong: Association for Computational Linguistics, Nov. 2019, pp. 24–28. doi: 10.18653/v1/D19-5904.

[46] M. Ochal, M. Patacchiola, A. Storkey, J. Vazquez, and S. Wang, "Few-Shot Learning with Class Imbalance," *IEEE Transactions on Artificial Intelligence*, vol. 4, no. 5, pp. 1348–1358, Jan. 2021, [Online]. Available: http://arxiv.org/abs/2101.02523

[47] S. Ravi and H. Larochelle, "OPTIMIZATION AS A MODEL FOR FEW-SHOT LEARNING," *ICLR*, pp. 1–10, 2017.

[48] A. Parnami and M. Lee, "Learning from Few Examples: A Summary of Approaches to Few-Shot Learning," *ArXiv*, pp. 1–31, Mar. 2022, [Online]. Available: http://arxiv.org/abs/2203.04291

[49] N. J. Prottasha *et al.*, "Transfer Learning for Sentiment Analysis Using BERT Based Supervised Fine-Tuning," *Sensors*, vol. 22, no. 11, Jun. 2022, doi: 10.3390/s22114157.

[50] F. Simon, "Fine-tuning," *Stanford Encyclopedia of Philosophy*, vol. 2, no. 4, pp. 1–25, Aug. 2017.

[51] S. Tian, L. Li, W. Li, H. Ran, X. Ning, and P. Tiwari, "A survey on few-shot class-incremental learning," Jan. 01, 2024, *Elsevier Ltd*. doi: 10.1016/j.neunet.2023.10.039.

[52] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a Few Examples: A Survey on Few-shot Learning," *ACM Comput Surv*, vol. 53, no. 3, Jun. 2020, doi: 10.1145/3386252.

[53] J. Dodge, G. Ilharco, R. Schwartz, A. Farhadi, H. Hajishirzi, and N. Smith, "Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping," *ArXiv*, pp. 1–10, Feb. 2020, [Online]. Available: http://arxiv.org/abs/2002.06305

[54] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A Survey on Deep Transfer Learning," *ArXiv*, vol. 2, no. 1, Aug. 2018, [Online]. Available: http://arxiv.org/abs/1808.01974

[55] G. Vrbančič and V. Podgorelec, "Transfer learning with adaptive fine-tuning," *IEEE Access*, vol. 8, no. 1, pp. 196197–196211, Nov. 2020, doi: 10.1109/ACCESS.2020.3034343.

[56] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ Questions for Machine Comprehension of Text," in *Proceedings of the 2016 Conference on Empirical*

Methods in Natural Language Processing, Texas, Nov. 2016, pp. 2383–2392. [Online]. Available: https://stanford-qa.com,

[57] C. P. Carrino, M. R. Costa-Jussà, and J. A. R. Fonollosa, "Automatic Spanish Translation of the SQuAD Dataset for Multilingual Question Answering," in *Proceedings of the 12th Conference on Language Resources and Evaluation*, Marseille, May 2020, pp. 5515–5523. [Online]. Available: https://github.com/robertostling/efmaral

[58] S. Wadhwa Khyathi Raghavi Chandu Eric Nyberg, "Comparative Analysis of Neural QA models on SQuAD," in *Proceedings of the Workshop on Machine Reading for Question Answering*, Melbourne, Jul. 2018, pp. 89–97. [Online]. Available: https://worksheets.codalab.org/worksheets/

[59] G. Sahu, P. Rodriguez, I. Laradji, P. Atighehchian, D. Vazquez, and D. Bahdanau, "Data Augmentation for Intent Classification with Off-the-shelf Large Language Models," in *Proceedings of the 4th Workshop on NLP for Conversational AI*, B. Liu, A. Papangelis, S. Ultes, A. Rastogi, Y.-N. Chen, G. Spithourakis, E. Nouri, and W. Shi, Eds., Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 47–57. doi: 10.18653/v1/2022.nlp4convai-1.5.

[60] S. Kobayashi, "Contextual Augmentation: Data Augmentation by Words with Paradigmatic Relations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, M. Walker, H. Ji, and A. Stent, Eds., New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 452–457. doi: 10.18653/v1/N18-2072.

[61] J. Berant and P. Liang, "Semantic Parsing via Paraphrasing," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, K. Toutanova and H. Wu, Eds., Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014, pp. 1415–1425. doi: 10.3115/v1/P14-1133.

[62] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *CoRR*, vol. abs/1412.6980, 2014, [Online]. Available: https://api.semanticscholar.org/CorpusID:6628106

[63] A. R. Aguilar, R. S. Moreno, L. Miranda, and W. Ojeda, "Algoritmo ADAM en la inteligencia artificial," in *Recuperado de https://www. riego. mx/congresos/comeii2021/files/ponencias/extenso/COMEII-21005. pdf*, 2021.

[64] A. Tamburrelli Giordano and Margara, "Towards Automated A/B Testing," in *Search-Based Software Engineering*, S. Le Goues Claire and Yoo, Ed., Cham: Springer International Publishing, 2014, pp. 184–198.

[65] A. Gilotte, C. Calauzènes, T. Nedelec, A. Abraham, and S. Dollé, "Offline A/B Testing for Recommender Systems," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, in WSDM '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 198–206. doi: 10.1145/3159652.3159687.

[66] J. Castaño, S. Martínez-Fernández, X. Franch, and J. Bogner, "Exploring the Carbon Footprint of Hugging Face's ML Models: A Repository Mining Study," in *2023 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, IEEE, Oct. 2023. doi: 10.1109/esem56168.2023.10304801.

[67] I. Challenger-Pérez, Y. D\'\iaz-Ricardo, and R. A. Becerra-Garc\'\ia, "El lenguaje de programación Python," *Ciencias Holgu\'\in*, vol. 20, no. 2, pp. 1–13, 2014.

[68] E. Stevens, L. Antiga, and T. Viehmann, *Deep learning with PyTorch*. Manning Publications, 2020.

[69] S. S. Sukhdeve Dr. Shitalkumar R. and Sukhdeve, “Google Colaboratory,” in *Google Cloud Platform for Data Science: A Crash Course on Big Data, Machine Learning, and Data Analytics Services*, Berkeley, CA: Apress, 2023, pp. 11–34. doi: 10.1007/978-1-4842-9688-2_2.

[70] T. Carneiro, R. V. Medeiros Da NóBrega, T. Nepomuceno, G.-B. Bian, V. H. C. De Albuquerque, and P. P. R. Filho, “Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications,” *IEEE Access*, vol. 6, pp. 61677–61685, 2018, doi: 10.1109/ACCESS.2018.2874767.

[71] M. J. Nelson and A. K. Hoover, “Notes on Using Google Colaboratory in AI Education,” in *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, in ITiCSE ’20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 533–534. doi: 10.1145/3341525.3393997.

[72] H. Kimm, I. Paik, and H. Kimm, “Performance Comparision of TPU, GPU, CPU on Google Colaboratory Over Distributed Deep Learning,” in *2021 IEEE 14th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, 2021, pp. 312–319. doi: 10.1109/MCSoC51149.2021.00053.

[73] J. V. Dillon *et al.*, “TensorFlow Distributions,” *ArXiv*, Nov. 2017, [Online]. Available: http://arxiv.org/abs/1711.10604

[74] P. Goldsborough, “A Tour of TensorFlow,” Oct. 2016, [Online]. Available: http://arxiv.org/abs/1610.01178

[75] B. Pang, E. Nijkamp, and Y. N. Wu, “Deep Learning With TensorFlow: A Review,” Apr. 01, 2020, *SAGE Publications Inc.* doi: 10.3102/1076998619872761.

[76] N. Ketkar, J. Moolayil, N. Ketkar, and J. Moolayil, “Introduction to pytorch,” *Deep learning with python: learn best practices of deep learning models with PyTorch*, pp. 27–91, 2021.

[77] S. Imambi, K. B. Prakash, and G. R. Kanagachidambaresan, “PyTorch,” *Programming with TensorFlow: Solution for Edge Computing Applications*, pp. 87–104, 2021.

[78] C.-Y. Lin, “ROUGE: A Package for Automatic Evaluation of Summaries,” *Association for Computational Linguistics*, pp. 74–81, Jul. 2004.

[79] O. Dušek and Z. Kasner, “Evaluating Semantic Accuracy of Data-to-Text Generation with Natural Language Inference,” in *Proceedings of the 13th International Conference on Natural Language Generation*, B. Davis, Y. Graham, J. Kelleher, and Y. Sripada, Eds., Dublin: Association for Computational Linguistics, Nov. 2020, pp. 131–137. doi: 10.18653/v1/2020.inlg-1.19.

[80] X. Sun and H. Wang, “Adjusting the Precision-Recall Trade-Off with Align-and-Predict Decoding for Grammatical Error Correction,” *Association for Computational Linguistics*, vol. 2, pp. 686–693, May 2022, doi: 10.18653/v1/2022.acl-short.77.

[81] R. Yacouby and D. Axman, “Probabilistic Extension of Precision, Recall, and F1 Score for More Thorough Evaluation of Classification Models,” in *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems*, S. Eger, Y. Gao, M. Peyrard, W. Zhao, and E. Hovy, Eds., Association for Computational Linguistics, Dec. 2020, pp. 79–91. doi: 10.18653/v1/2020.eval4nlp-1.9.

[82] J.-P. Ng and V. Abrecht, “Better Summarization Evaluation with Word Embeddings for ROUGE,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisboa: Association for Computational Linguistics, Sep. 2015, pp. 1925–1930. doi: 10.18653/v1/D15-1222.

[83] I. Kolyshkina and S. Simoff, “Interpretability of Machine Learning Solutions in Industrial Decision Engineering,” in *Data Mining*, T. D. Le, K.-L. Ong, Y. Zhao, W. H. Jin, S. Wong,

L. Liu, and G. Williams, Eds., Springer Singapore, 2019, pp. 156–170. doi: 10.1007/978-981-15-1699-3_13.

[84] D. MacQueen, R. Harper, and J. Reppy, “The history of Standard ML,” *Proc. ACM Program. Lang.*, vol. 4, no. HOPL, Jun. 2020, doi: 10.1145/3386336.

[85] T. Zhang *et al.*, “Operationalizing AI in Future Networks: A Bird’s Eye View from the System Perspective,” *CoRR*, vol. abs/2303.04073, 2023, doi: 10.48550/ARXIV.2303.04073.

[86] M. M. John, H. H. Olsson, and J. Bosch, “Developing ML/DL Models: A Design Framework,” in *Proceedings of the International Conference on Software and System Processes*, in ICSSP ’20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 1–10. doi: 10.1145/3379177.3388892.

[87] I. Kolyshkina and S. Simoff, “Interpretability of machine learning solutions in public healthcare: The CRISP-ML approach,” *Front Big Data*, vol. 4, p. 660206, 2021.

[88] V. Sahana *et al.*, “The DistilBERT Model: A Promising Approach to Improve Machine Reading Comprehension Models,” *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 11, no. 8, pp. 293–309, Aug. 2023, doi: 10.17762/ijritcc.v11i8.7957.

[89] Y. Boreshban, S. M. Mirbostani, G. Ghassem-Sani, S. A. Mirroshandel, and S. Amiriparian, “Improving Question Answering Performance Using Knowledge Distillation and Active Learning,” *Eng Appl Artif Intell*, vol. 123, no. 3, Sep. 2021, doi: 10.1016/j.engappai.2023.106137.

[90] S. S. Lakkimsetty, S. V. Latchireddy, S. M. Lakkoju, G. R. Manukonda, and R. V. V. M. Krishna, “Fine-Tuned Transformer Models for Question Answering,” in *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2023, pp. 1–5.

[91] M. Popel and O. Bojar, “Training Tips for the Transformer Model,” *The Prague Bulletin of Mathematical Linguistics*, vol. 110, no. 1, pp. 43–70, Apr. 2018, doi: 10.2478/pralin-2018-0002.

[92] A. Chen, G. Stanovsky, S. Singh, and M. Gardner, “Evaluating Question Answering Evaluation,” in *Proceedings ofthe Second Workshop on Machine Reading for Question Answering*, Hong Kong: Association for Computational Linguistics, Nov. 2019, pp. 119–124. doi: 10.18653/v1/D19-5817.

11. Anexos

Anexo 1. Requisitos del proyecto de vinculación “Inclusión lectora de estudiantes con discapacidad visual mediante innovación tecnológica”

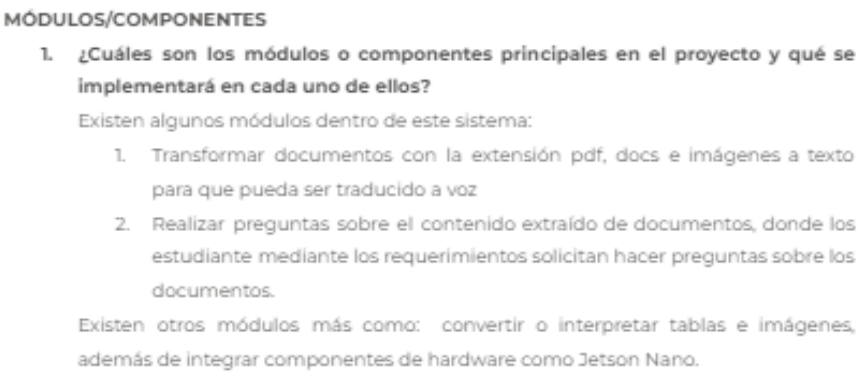
4.1.5 Buscar en el documento

Historia de Usuario	
Número: 5	**Nombre:** Buscar en el documento
Usuario: Usuario	**Iteración asignada:** 5
Prioridad en negocio: Alta	**Puntos Estimados:** 1.5
Riesgo en desarrollo: Alta	**Puntos Reales:** 1
Descripción: Deseo que la aplicación permita realizar búsqueda dentro del documento.	
Observaciones:	

Figura 35. *Requisito principal para elaborar el modelo QA*

Ver documento completo de requisitos en el repositorio github

Anexo 2. Entrevista al Ing. Oscar Cumbicus director del proyecto de Vinculación de la Carrera de Computación



MÓDULOS/COMPONENTES

1. **¿Cuáles son los módulos o componentes principales en el proyecto y qué se implementará en cada uno de ellos?**

   Existen algunos módulos dentro de este sistema:

   1. Transformar documentos con la extensión pdf, docs e imágenes a texto para que pueda ser traducido a voz
   2. Realizar preguntas sobre el contenido extraído de documentos, donde los estudiante mediante los requerimientos solicitan hacer preguntas sobre los documentos.

   Existen otros módulos más como: convertir o interpretar tablas e imágenes, además de integrar componentes de hardware como Jetson Nano.

Figura 36. *Pregunta principal de la entrevista*

Ver entrevista completa en el repositorio github

Anexo 3. Obtención búsqueda de documentos

El principal paso es seleccionar las bases de datos, indexadores, entre otras fuentes de búsqueda de información. En la **Tabla 21** se muestra las principales plataformas usadas para obtener los documentos que argumentan esta TIC.

Tabla 21. *Principales plataformas usadas para la búsqueda de documentos*

Plataformas	
Nombre	**Link de acceso**
ACM Digital Library	https://dl.acm.org
ACL Anthology	https://aclanthology.org
IEEE Digital Library	https://ieeexplore.ieee.org/Xplore/home.jsp
Google Scholar	https://scholar.google.com
Scopus	https://www.scopus.com
Springer	https://link.springer.com
arXiv	https://arxiv.org

El siguiente paso que se realiza para la búsqueda de información es obtener las palabras clave que se encuentran en los objetivos, el tema o la hipótesis de cualquier Trabajo de Integración curricular, muchas de las veces se encuentran en forma de variables: Question Answering model, SQuAD, Fine-Tuning, DistilBERT y ROUGE. Como los últimos pasos es redactar las cadenas de búsqueda, y establecer los criterios de inclusión y exclusión para elegir rechazar los artículos, documentos, libros, entre otros trabajos que se obtienen luego de aplicar las cadenas de búsqueda. En la **Tabla 22** se muestra las cadenas de búsqueda establecidas y en la **Tabla 23** se muestra los criterios de inclusión y exclusión.

Tabla 22. *Cadenas de Búsqueda*

Repositorio Virtual	Cadena de búsqueda
ACM Digital Library	[[Publication Title: distilbert] OR [Publication Title: squad]] AND [All: performance of machine learning models] AND [Keywords: distilbert] AND [Abstract: answers to questions]
IEEE Digital Library	"DistilBERT" AND "SQuAD" AND "fine-tuning" AND "ROUGE" AND "question answering model" AND "performance of machine learning models"
Google Scholar	"DistilBERT" AND "SQuAD" AND "fine-tuning" AND "ROUGE" AND "question answering model" AND "performance of machine learning models"
Scopus	(TITLE-ABS-KEY (distilbert AND question AND answering AND model) OR TITLE-ABS-KEY (squad) AND TITLE-ABS-KEY (fine-tuning AND ROUGE) OR TITLE-ABS-KEY (performance AND of AND distilbert) AND TITLE-ABS-KEY (academic AND content)) AND (LIMIT-TO (PUBYEAR , 2021) OR LIMIT-TO (PUBYEAR , 2020) OR LIMIT-TO (PUBYEAR , 2019))
Springer	"DistilBERT" AND "SQuAD" AND "fine-tuning" AND "ROUGE" AND "question answering model" AND "performance of machine learning models"
arXiv	"DistilBERT" AND "SQuAD" AND "fine-tuning" AND "ROUGE" AND "question answering model" AND "performance of machine learning models"
ACL Anthology	"DistilBERT" AND "SQuAD" AND "fine-tuning" AND "ROUGE" AND "question answering model" AND "performance of machine learning models"

Tabla 23. *Criterios de inclusión y exclusión*

Criterios de Inclusión y Exclusión	
Inclusión	**Exclusión**
• Artículos que investiguen el ajuste de un modelo QA mediante Fine Tuning en DistilBERT. • Estudios que evalúen la puntuación ROUGE como métrica del rendimiento del modelo QA. • Investigaciones centradas en responder preguntas sobre el contenido extraído de textos. • Que se encuentre en idioma inglés o español • Que se encuentren en revistas de alto impacto • Artículos de acceso abierto	• Artículos de antecedentes (background articles). • Resultados incorrectos. • Tipos de publicación incorrectos (solo necesitamos libros, revistas y tesis). • Estudios que no se encuentren en el rango de los 5 últimos años (excepción en casos especiales). • Documentos que no sea reproducibles. • Artículos que no tengan conclusiones sobre el tema. • Artículos que no tengan una metodología. • Documentos que tengan bibliografía, citas y referencias falsas.

Anexo 4. *Códigos de cada uno de los 4 experimentos en el ajuste de un modelo QA*

Para entender cómo se realiza el ajuste debe leer todo el código, aunque todos los códigos tienen una estructura similar por lo que basta con entender un modelo. En la **Figura 37**, **Figura 38**, **Figura 39** y **Figura 40** se puede evidenciar las primeras líneas de código usadas en cada uno de los experimentos realizados.

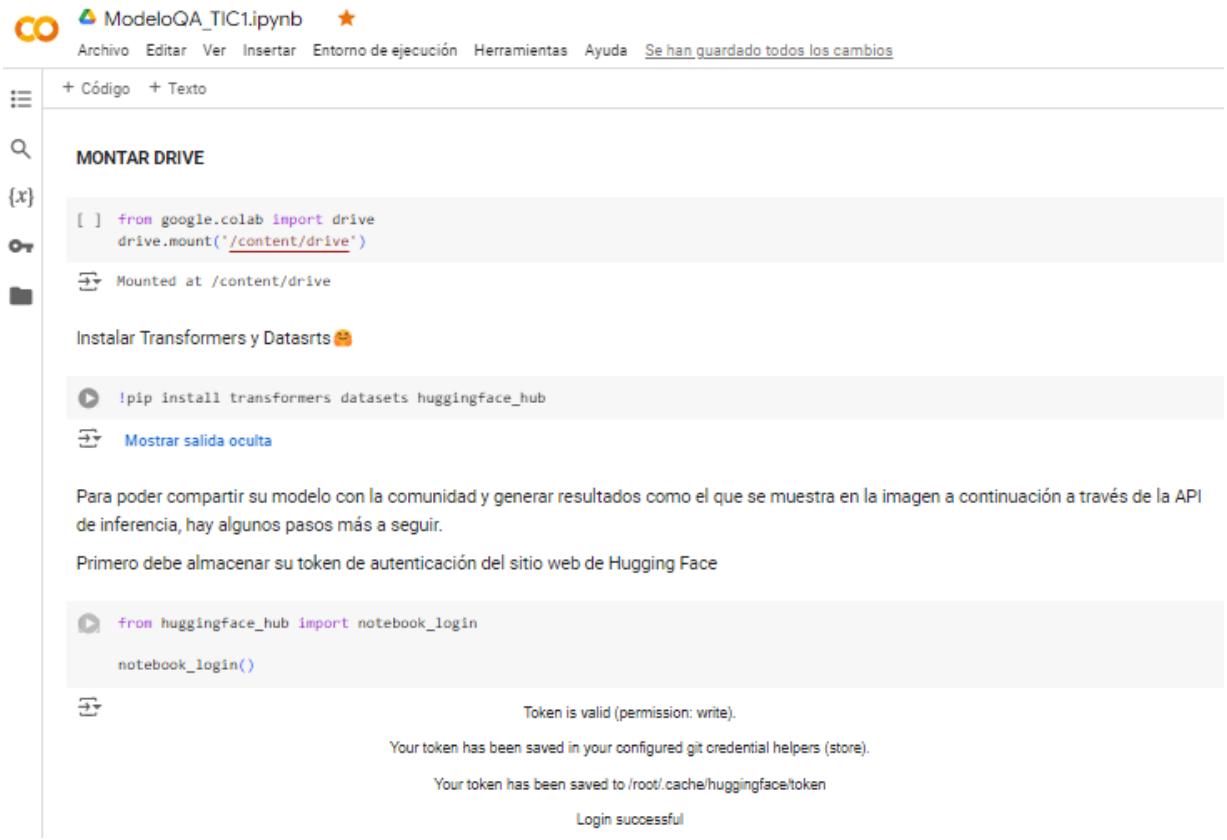


Figura 37. *Código llamado ModeloQA_TIC1.ipynb, correspondiente al primer experimento*

Ver código completo en drive.

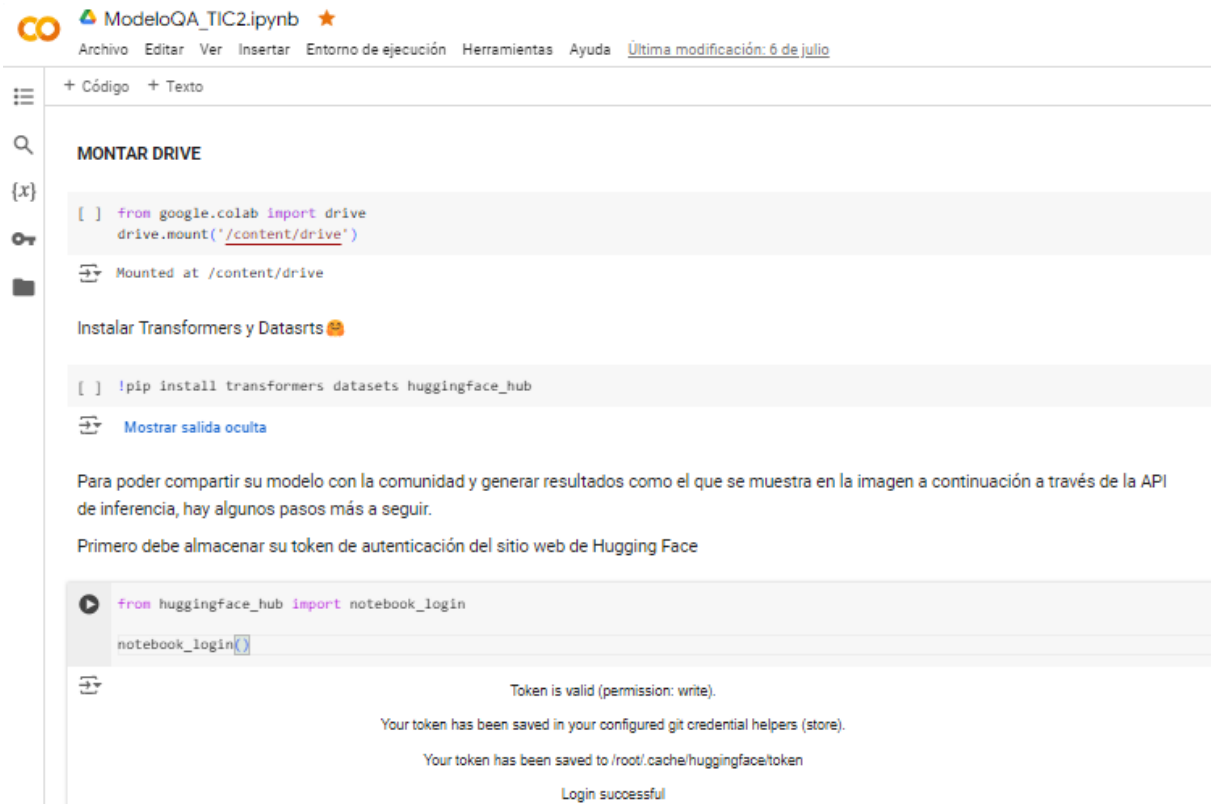


Figura 38. *Código llamado ModeloQA_TIC2.ipynb, correspondiente al segundo experimento*

Ver código completo en drive

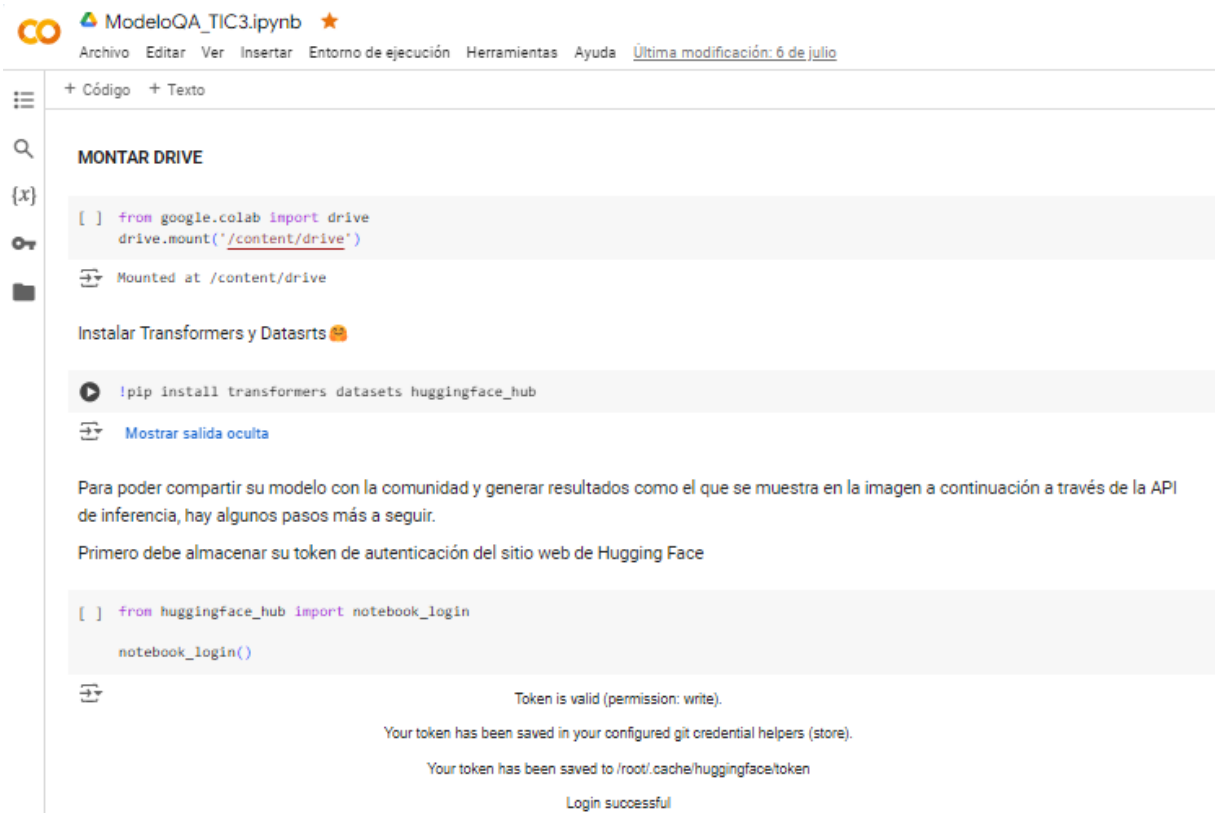


Figura 39. *Código llamado ModeloQA_TIC3.ipynb, correspondiente al tercer experimento*

Ver código completo en drive

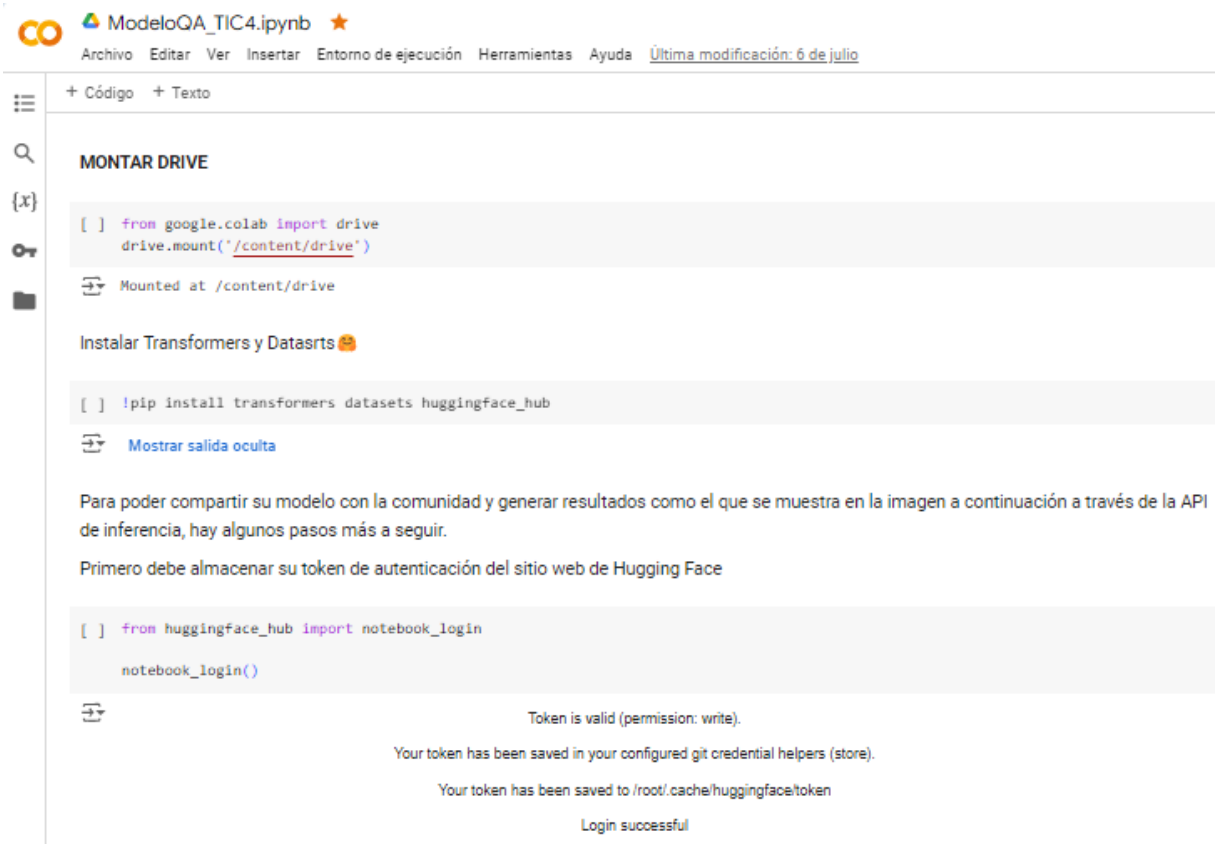


Figura 40. *Código llamado ModeloQA_TIC4.ipynb, correspondiente al cuarto experimento*

Ver código completo en drive

Anexo 5. Estadísticas durante la etapa de entrenamiento de los 4 experimentos:

Train Loss	Train End Logits Accuracy	Train Start Logits Accuracy	Validation Loss	Validation End Logits Accuracy	Validation Start Logits Accuracy	Epoch
2.9391	0.3582	0.3183	1.5236	0.5611	0.5862	0
1.1532	0.6292	0.6586	0.8487	0.7085	0.7179	1
0.7185	0.7521	0.7563	0.7432	0.7962	0.7555	2
0.6133	0.7763	0.7784	0.6925	0.7712	0.7524	3
0.4777	0.8288	0.8267	0.6963	0.7524	0.7962	4
0.4441	0.8298	0.8483	0.6422	0.8182	0.7886	5
0.3896	0.8519	0.8645	0.6378	0.7908	0.8856	6
0.3642	0.8508	0.8771	0.6286	0.8088	0.7994	7
0.3068	0.8960	0.8887	0.5387	0.8433	0.8558	8
0.2755	0.8845	0.8845	0.6849	0.8245	0.8387	9
0.2711	0.9023	0.9023	0.5653	0.8527	0.8370	10
0.2268	0.9065	0.9282	0.6267	0.8589	0.8150	11
0.2016	0.9086	0.9286	0.6835	0.8777	0.8481	12
0.2044	0.9107	0.9296	0.5385	0.8840	0.8683	13
0.1923	0.9275	0.9296	0.5440	0.8871	0.8621	14
0.1448	0.9307	0.9496	0.5563	0.8934	0.8464	15
0.1465	0.9359	0.9454	0.5626	0.8809	0.8589	16
0.1323	0.9464	0.9517	0.6286	0.8401	0.8495	17
0.1291	0.9506	0.9586	0.5277	0.8746	0.8621	18
0.1156	0.9590	0.9559	0.5341	0.8777	0.8558	19
0.0839	0.9643	0.9748	0.5753	0.8903	0.8527	20
0.1087	0.9538	0.9653	0.5299	0.8746	0.8558	21
0.0981	0.9664	0.9664	0.6834	0.8558	0.8464	22
0.0791	0.9716	0.9779	0.6137	0.8777	0.8495	23
0.0782	0.9653	0.9748	0.6260	0.8809	0.8589	24
0.0747	0.9748	0.9748	0.5973	0.8903	0.8527	25
0.0685	0.9653	0.9821	0.6887	0.8809	0.8777	26
0.0688	0.9685	0.9737	0.5546	0.8903	0.8495	27
0.0513	0.9811	0.9842	0.5925	0.8997	0.8495	28
0.0518	0.9769	0.9863	0.6222	0.8777	0.8746	29
0.0451	0.9748	0.9916	0.6382	0.8777	0.8746	30
0.0424	0.9842	0.9811	0.6389	0.8871	0.8652	31
0.0392	0.9800	0.9853	0.6361	0.8809	0.8715	32
0.0382	0.9842	0.9895	0.6253	0.8840	0.8715	33
0.0485	0.9800	0.9905	0.6734	0.8715	0.8777	34
0.0485	0.9769	0.9905	0.6184	0.8903	0.8652	35
0.0364	0.9790	0.9926	0.6584	0.8809	0.8715	36
0.0272	0.9842	0.9947	0.6439	0.8871	0.8715	37
0.0248	0.9916	0.9937	0.6390	0.8934	0.8746	38
0.0211	0.9884	0.9958	0.6597	0.8871	0.8683	39
0.0277	0.9916	0.9926	0.6561	0.8809	0.8683	40
0.0307	0.9884	0.9874	0.6669	0.8809	0.8652	41
0.0186	0.9947	0.9947	0.6526	0.8871	0.8652	42
0.0178	0.9905	0.9958	0.6681	0.8840	0.8621	43
0.0195	0.9905	0.9926	0.6780	0.8903	0.8683	44
0.0197	0.9937	0.9916	0.7142	0.8777	0.8558	45
0.0176	0.9947	0.9926	0.6914	0.8809	0.8715	46
0.0188	0.9947	0.9916	0.6981	0.8809	0.8652	47
0.0158	0.9958	0.9926	0.6845	0.8809	0.8715	48
0.0178	0.9937	0.9905	0.6826	0.8840	0.8715	49
0.0173	0.9926	0.9937	0.6833	0.8809	0.8746	50

Figura 41. *Estadísticas durante el entrenamiento del distilbert-base-uncased-QA1-finetuned-squad-es*

Ver código completo en Hugging Face

Train Loss	Train End Logits Accuracy	Train Start Logits Accuracy	Validation Loss	Validation End Logits Accuracy	Validation Start Logits Accuracy	Epoch
2.3428	0.4160	0.4317	1.3438	0.5611	0.6458	0
1.1526	0.6261	0.6397	1.0597	0.6677	0.7429	1
0.7612	0.7269	0.7647	1.0245	0.7210	0.7806	2
0.5528	0.7836	0.8319	1.2436	0.7116	0.7712	3
0.4667	0.8340	0.8435	1.0705	0.7524	0.7555	4
0.3834	0.8813	0.8687	1.1209	0.7586	0.7712	5
0.3678	0.8634	0.8876	1.2341	0.7618	0.7649	6
0.2555	0.9044	0.9181	1.1561	0.7649	0.8056	7
0.2151	0.9160	0.9328	1.0908	0.7931	0.7994	8
0.1855	0.9256	0.9475	1.2809	0.7994	0.7774	9
0.1654	0.9443	0.9454	1.3974	0.7837	0.7806	10
0.1282	0.9464	0.9517	1.4260	0.7774	0.7837	11
0.1313	0.9443	0.9601	1.4537	0.7900	0.7962	12
0.1301	0.9517	0.9590	1.1851	0.7774	0.8150	13
0.1089	0.9548	0.9590	1.2442	0.7774	0.8088	14
0.1023	0.9601	0.9622	1.4575	0.7931	0.7931	15
0.0956	0.9590	0.9685	1.5160	0.7837	0.7900	16
0.0712	0.9727	0.9737	1.5741	0.7900	0.8088	17
0.0752	0.9674	0.9790	1.4401	0.7931	0.7994	18
0.0604	0.9737	0.9779	1.6410	0.7962	0.8088	19
0.0497	0.9758	0.9821	1.5655	0.7962	0.8119	20
0.0668	0.9685	0.9811	1.3480	0.7806	0.7962	21
0.0567	0.9769	0.9800	1.3820	0.7900	0.8088	22
0.0550	0.9769	0.9832	1.3593	0.7806	0.8056	23
0.0399	0.9821	0.9884	1.5254	0.7868	0.7931	24
0.0320	0.9842	0.9874	1.5801	0.7868	0.7994	25
0.0296	0.9852	0.9884	1.6310	0.7962	0.7962	26
0.0307	0.9863	0.9926	1.4756	0.7774	0.7900	27
0.0254	0.9863	0.9895	1.7564	0.7774	0.7931	28
0.0255	0.9853	0.9937	1.6061	0.7774	0.7962	29
0.0214	0.9863	0.9937	1.7697	0.7712	0.8056	30
0.0283	0.9842	0.9863	1.8398	0.7806	0.7900	31
0.0182	0.9905	0.9926	1.8756	0.7837	0.7994	32
0.0252	0.9832	0.9947	1.8182	0.7837	0.7962	33
0.0222	0.9863	0.9947	1.7854	0.7837	0.7931	34
0.0216	0.9884	0.9947	1.5707	0.7931	0.8025	35
0.0161	0.9937	0.9916	1.7071	0.7806	0.8025	36
0.0146	0.9926	0.9926	1.7827	0.7868	0.7962	37
0.0148	0.9905	0.9947	1.8678	0.7868	0.7931	38
0.0117	0.9884	0.9968	1.7944	0.7868	0.7900	39
0.0137	0.9905	0.9958	1.7666	0.7900	0.7931	40
0.0160	0.9874	0.9958	1.7644	0.7868	0.7962	41
0.0150	0.9916	0.9937	1.7783	0.7868	0.8025	42
0.0128	0.9895	0.9958	1.7480	0.7900	0.7994	43
0.0102	0.9937	0.9947	1.7432	0.7931	0.7994	44
0.0138	0.9947	1.0	1.7511	0.7931	0.7994	45

Figura 42. *Estadísticas durante el entrenamiento del distilbert-base-uncased-QA2-finetuned-squad-es*

Ver código completo en Hugging Face

Train Loss	Train End Logits Accuracy	Train Start Logits Accuracy	Validation Loss	Validation End Logits Accuracy	Validation Start Logits Accuracy	Epoch
4.7467	0.1006	0.0561	5.8046	0.0157	0.0578	0
4.8045	0.0148	0.0138	5.2042	0.0094	0.0094	1
5.9402	0.0032	0.0053	5.9506	0.0031	0.0063	2
5.9626	0.0021	0.0021	5.9506	0.0031	0.0031	3
5.9599	0.0042	0.0	5.9506	0.0	0.0	4
5.9718	0.0	0.0011	5.9506	0.0	0.0031	5
5.9587	0.0021	0.0064	5.9506	0.0031	0.0031	6
5.9657	0.0064	0.0032	5.9506	0.0031	0.0188	7
5.9617	0.0021	0.0032	5.9506	0.0031	0.0063	8
5.9596	0.0021	0.0032	5.9506	0.0	0.0031	9
5.9648	0.0021	0.0021	5.9506	0.0094	0.0063	10
5.9608	0.0021	0.0032	5.9506	0.0125	0.0094	11
5.9567	0.0021	0.0053	5.9506	0.0063	0.0	12
5.9625	0.0011	0.0011	5.9506	0.0	0.0	13
5.9640	0.0	0.0011	5.9506	0.0031	0.0	14
5.9606	0.0011	0.0	5.9506	0.0063	0.0063	15
5.9622	0.0032	0.0053	5.9506	0.0094	0.0063	16
5.9600	0.0011	0.0021	5.9506	0.0	0.0063	17
5.9579	0.0011	0.0011	5.9506	0.0063	0.0094	18
5.9598	0.0032	0.0053	5.9506	0.0031	0.0	19
5.9589	0.0021	0.0032	5.9506	0.0063	0.0031	20
5.9566	0.0032	0.0021	5.9506	0.0	0.0	21
5.9536	0.0011	0.0053	5.9506	0.0	0.0	22
5.9592	0.0021	0.0021	5.9506	0.0031	0.0031	23
5.9548	0.0032	0.0042	5.9506	0.0	0.0	24
5.9569	0.0	0.0021	5.9506	0.0	0.0	25
5.9640	0.0032	0.0011	5.9506	0.0031	0.0031	26
5.9497	0.0011	0.0011	5.9506	0.0	0.0031	27
5.9558	0.0	0.0053	5.9506	0.0063	0.0031	28
5.9563	0.0021	0.0032	5.9506	0.0063	0.0063	29
5.9585	0.0032	0.0032	5.9506	0.0	0.0094	30
5.9569	0.0011	0.0021	5.9506	0.0094	0.0063	31
5.9580	0.0011	0.0021	5.9506	0.0063	0.0	32
5.9532	0.0032	0.0011	5.9506	0.0	0.0063	33
5.9525	0.0021	0.0032	5.9506	0.0	0.0	34
5.9552	0.0042	0.0011	5.9506	0.0	0.0	35
5.9538	0.0021	0.0032	5.9506	0.0	0.0	36
5.9538	0.0032	0.0032	5.9506	0.0031	0.0063	37
5.9567	0.0011	0.0021	5.9506	0.0063	0.0031	38
5.9570	0.0053	0.0032	5.9506	0.0	0.0031	39
5.9545	0.0032	0.0	5.9506	0.0	0.0063	40

Figura 43. *Estadísticas durante el entrenamiento del distilbert-base-uncased-QA3-finetuned-squad-es*

Ver código completo en Hugging Face

Train Loss	Train End Logits Accuracy	Train Start Logits Accuracy	Validation Loss	Validation End Logits Accuracy	Validation Start Logits Accuracy	Epoch
3.5949	0.1733	0.1891	2.4981	0.3918	0.3981	0
2.0479	0.4097	0.4811	1.6575	0.4890	0.6113	1
1.4343	0.5599	0.6166	1.3371	0.5768	0.6426	2
1.0892	0.6313	0.6891	1.1850	0.6677	0.6865	3
0.9172	0.6870	0.7405	1.1305	0.6771	0.7335	4
0.7470	0.7255	0.7910	1.0674	0.7147	0.7524	5
0.6728	0.7426	0.8088	1.0843	0.7116	0.7680	6
0.5989	0.7721	0.8403	1.0787	0.7304	0.7649	7
0.4988	0.8057	0.8582	1.1091	0.7398	0.7618	8
0.4674	0.8214	0.8540	1.1150	0.7367	0.7774	9
0.4173	0.8256	0.8782	1.1434	0.7335	0.7774	10
0.3804	0.8319	0.8897	1.1256	0.7335	0.7900	11
0.3831	0.8456	0.8834	1.1614	0.7429	0.7931	12
0.3325	0.8550	0.9097	1.1519	0.7429	0.7900	13
0.3115	0.8739	0.9076	1.1423	0.7586	0.7868	14
0.2860	0.8792	0.9160	1.1335	0.7649	0.8025	15
0.2751	0.8834	0.9181	1.1135	0.7712	0.8119	16
0.2441	0.8918	0.9296	1.1771	0.7524	0.7900	17
0.2342	0.9044	0.9370	1.1433	0.7680	0.8088	18
0.2049	0.9254	0.9391	1.1689	0.7680	0.7994	19
0.2029	0.9170	0.9475	1.1659	0.8025	0.8150	20
0.1939	0.9170	0.9422	1.2030	0.7712	0.8150	21
0.1787	0.9202	0.9548	1.2073	0.7806	0.8056	22
0.2013	0.9233	0.9485	1.1615	0.7962	0.7994	23
0.1821	0.9349	0.9443	1.1657	0.7806	0.8088	24
0.1683	0.9328	0.9464	1.1684	0.7994	0.8088	25
0.1568	0.9286	0.9580	1.1909	0.7900	0.8056	26
0.1536	0.9244	0.9590	1.2054	0.7868	0.8182	27
0.1221	0.9485	0.9601	1.1996	0.7806	0.8088	28
0.1373	0.9349	0.9601	1.2201	0.7806	0.8056	29
0.1334	0.9443	0.9569	1.2531	0.7868	0.8025	30
0.1335	0.9422	0.9569	1.2030	0.7962	0.8088	31
0.1157	0.9485	0.9590	1.2142	0.7931	0.8088	32
0.1209	0.9475	0.9590	1.2215	0.7743	0.7994	33
0.1149	0.9548	0.9653	1.2125	0.7806	0.8056	34
0.1048	0.9538	0.9674	1.2632	0.7900	0.8056	35
0.1056	0.9475	0.9706	1.2485	0.7931	0.8088	36
0.0964	0.9653	0.9685	1.2468	0.7900	0.8088	37
0.1000	0.9559	0.9664	1.2422	0.7962	0.8056	38
0.0989	0.9601	0.9653	1.2620	0.8025	0.8056	39
0.1024	0.9590	0.9674	1.2528	0.7994	0.8056	40
0.0917	0.9548	0.9716	1.2506	0.7931	0.8088	41
0.0913	0.9580	0.9685	1.2538	0.8025	0.8056	42
0.0923	0.9664	0.9632	1.2619	0.8025	0.8056	43
0.0921	0.9559	0.9643	1.2621	0.8056	0.8088	44
0.0931	0.9559	0.9685	1.2632	0.8088	0.8088	45

Figura 44. *Estadísticas durante el entrenamiento del distilbert-base-uncased-QA4-finetuned-squad-es*

Ver código completo en [Hugging Face]

Anexo 6. Gráficos estadísticos correspondientes al entrenamiento del modelo QA

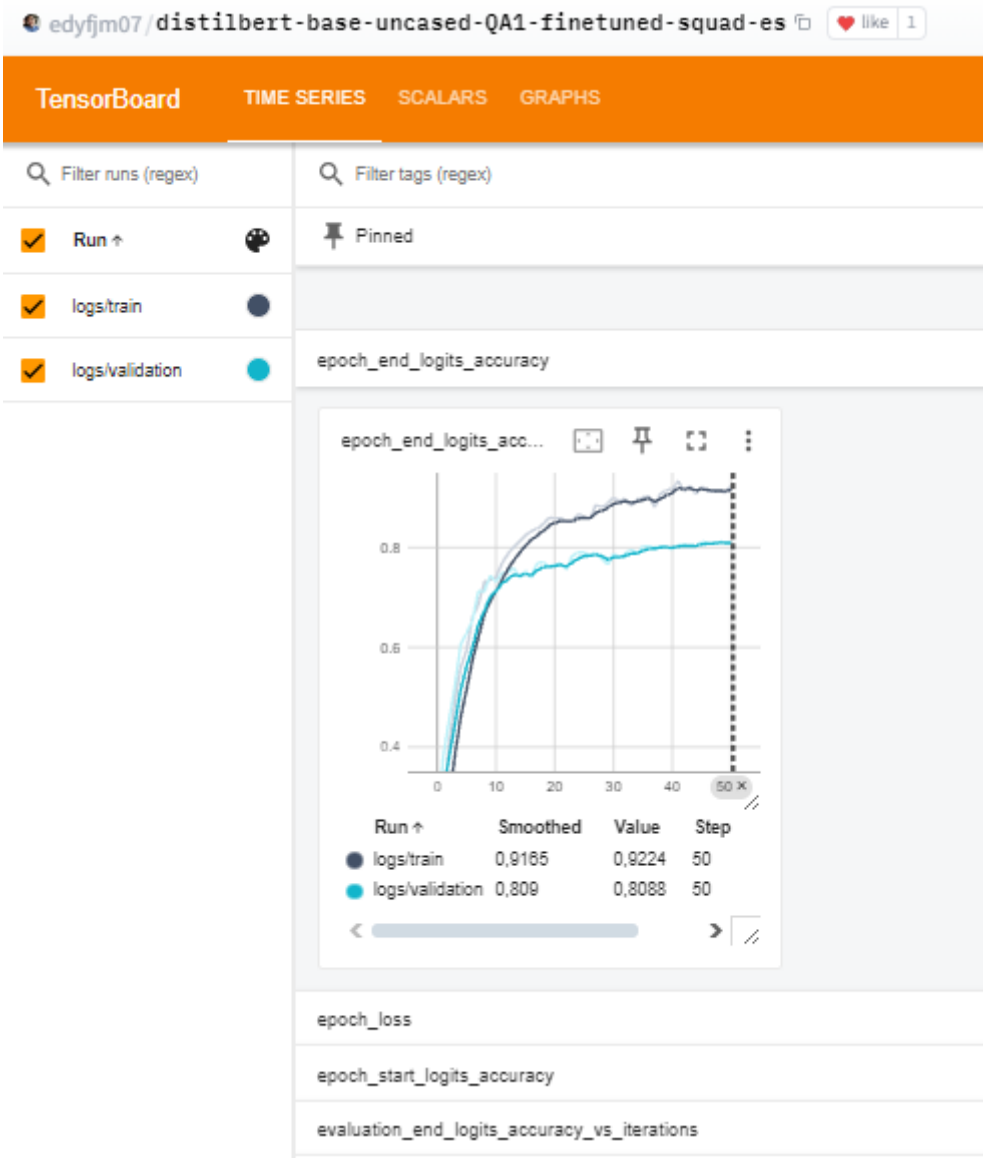


Figura 45. *Gráficos estadísticos del modelo distilbert-base-uncased-QA1-finetuned-squad-es*

Ver estadísticas completas y específicas de cada elemento en Hugging Face

Anexo 7. Modelo distilbert-base-uncased-QA1-finetuned-squad-es desplegado en Hugging Face

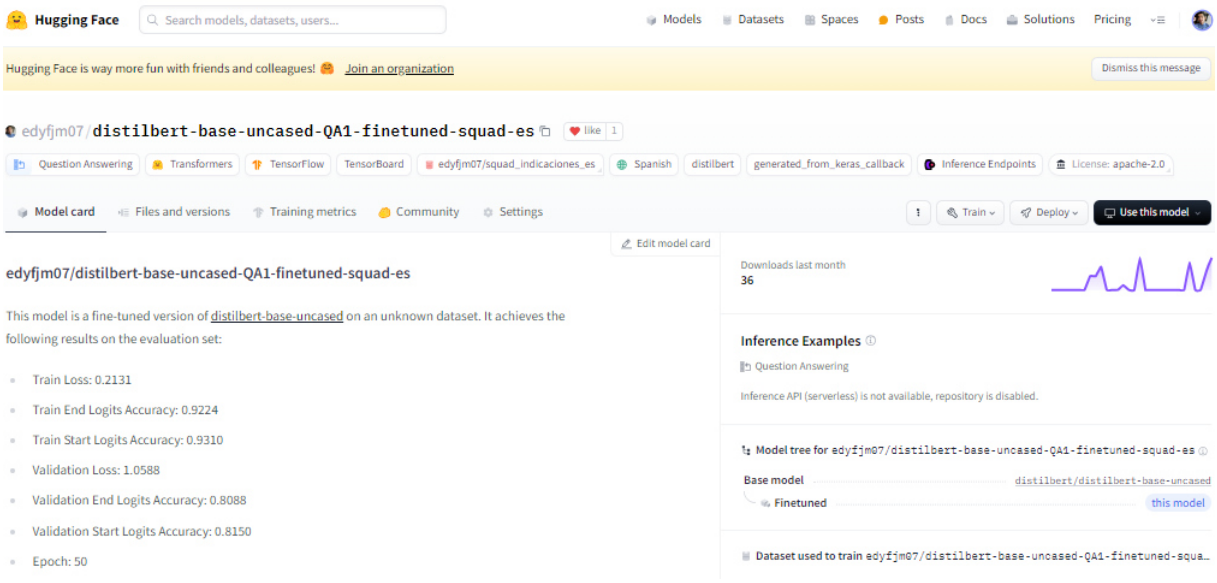


Figura 46. *Modelo QA desplegado en la plataforma Hugging Face*

Anexo 8. Cálculo individual de la métrica ROUGE de forma automática

Tabla 24. *Cálculo unitario de ROUGE en los datos del test*

Datos sobre el cálculo de la métrica ROUGE de manera Automática				
Contexto	**Pregunta**	**Respuesta de referencia**	**Respuesta del modelo**	**Valor de ROUGE**
Expansión y personalización: Si lo deseas, puedes expandir aún más el mapa mental agregando subramas o detalles adicionales a medida que exploras cada tema en más profundidad. Personaliza el mapa mental de acuerdo con tus propias experiencias y necesidades como estudiante universitario.	¿Cómo se puede expandir aún más el mapa mental?	Agregando subramas o detalles adicionales	agregando subramas o detalles adicionales	{'rouge1': AggregateScore(low=Score(precision=1.0, recall=1.0, fmeasure=1.0), mid=Score(precision=1.0, recall=1.0, fmeasure=1.0), high=Score(precision=1.0, recall=1.0, fmeasure=1.0)), 'rouge2': AggregateScore(low=Score(precision=1.0, recall=1.0, fmeasure=1.0), mid=Score(precision=1.0, recall=1.0, fmeasure=1.0), high=Score(precision=1.0, recall=1.0, fmeasure=1.0)), 'rougeL': AggregateScore(low=Score(precision=1.0, recall=1.0, fmeasure=1.0), mid=Score(precision=1.0, recall=1.0, fmeasure=1.0), high=Score(precision=1.0, recall=1.0, fmeasure=1.0)), 'rougeLsum': AggregateScore(low=Score(precision=1.0, recall=1.0, fmeasure=1.0), mid=Score(precision=1.0, recall=1.0, fmeasure=1.0), high=Score(precision=1.0, recall=1.0, fmeasure=1.0))}

Nota: Para ver todos los valores ingrese al repositorio de drive

Anexo 9. *Gráficos adicionales de la métrica ROUGE*

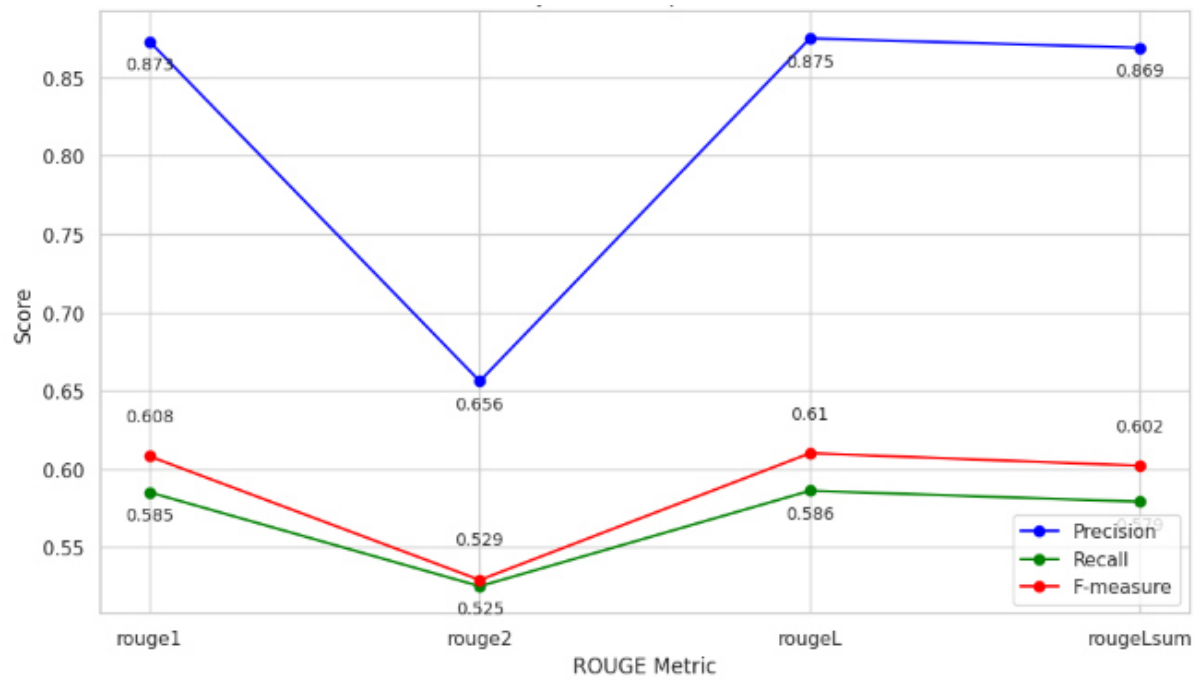


Figura 47. *Gráfico de líneas respecto a la métrica ROUGE*

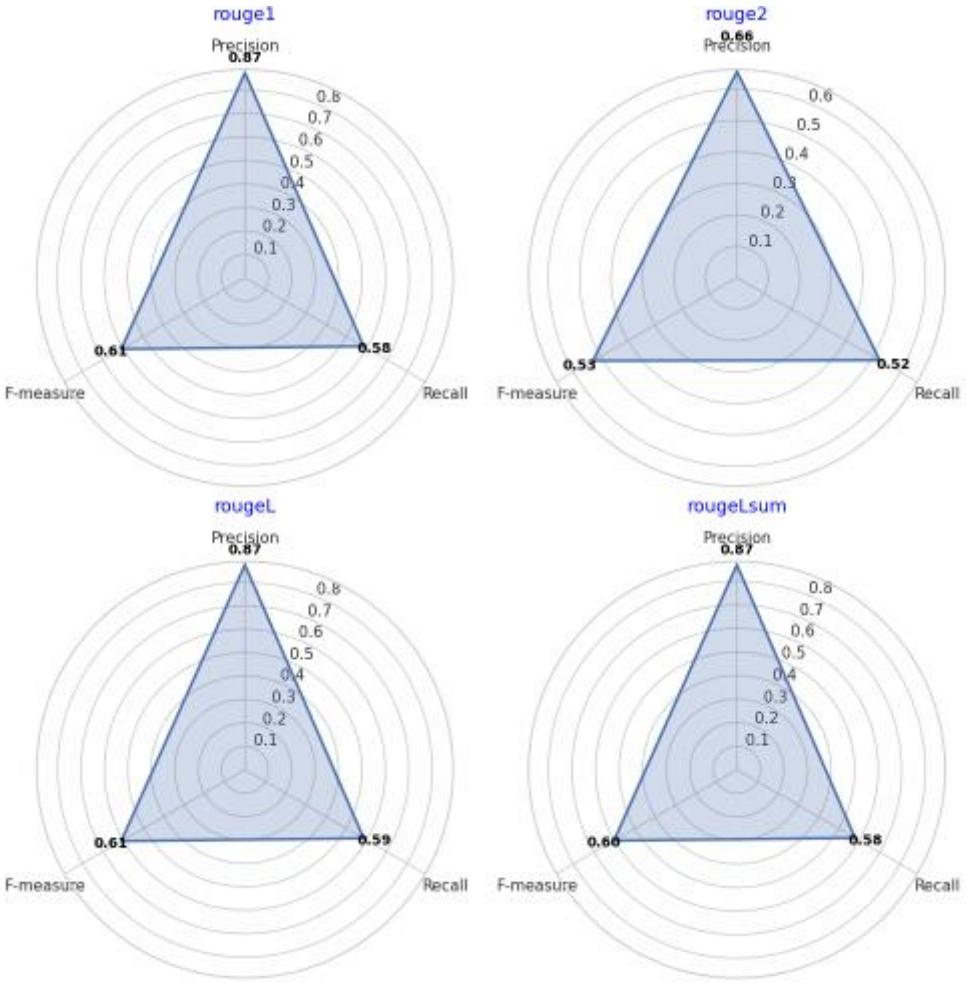


Figura 48. *Gráfico radar sobre la métrica ROUGE*

Anexo 10. Validación y aprobación del modelo QA

Certificado de validación del modelo QA

Loja, 20 de septiembre de 2024

Oscar Miguel Cumbicus Pineda, Mg.Sc.

Especialista en Procesamiento de Lenguaje Natural

CERTIFICO:

Que después de una evaluación detallada, el modelo de Question Answering (QA) desarrollado por el estudiante **Edy Francisco Jiménez Merino** con cédula de identidad **Nro. 1950170389**, resultó ser funcional y eficiente, con resultados modestos. El modelo ha mostrado precisión y exactitud adecuadas en la extracción de respuestas, manejando eficientemente la variabilidad en las preguntas sobre contenido extraído de tareas y documentos académicos de la Carrera de computación de la Universidad Nacional de Loja. A pesar de que las métricas de evaluación no alcanzaron niveles excepcionales, los resultados **son suficientes para cumplir con los objetivos del proyecto**.

Firmado electrónicamente por:
OSCAR MIGUEL
CUMBICUS PINEDA

Ing. Oscar Miguel Cumbicus Pineda, Mg.Sc.

Especialista en Procesamiento de Lenguaje Natural

Anexo 11. Certificado de traducción del resumen

Loja, 02 de agosto de 2024

Lic. Pedro Geovanny Calva Jiménez
LICENCIADO EN PEDAGOGÍA DEL IDIOMA INGLÉS

CERTIFICO:

Que el resumen del Trabajo de Integración Curricular cuyo título es: **Modelo QA basado en DistilBERT para responder a preguntas sobre el contenido extraído de tareas académicas de la carrera de Computación de la UNL**, del aspirante **Edy Francisco Jiménez Merino**, con cédula de identidad Nro. **1950170389**, de la Carrera de Computación de la Universidad Nacional de Loja, ha sido traducido al inglés y cumple con las características propias del idioma extranjero.

Lo certifico en honor a la verdad y autorizo hacer uso del presente en lo que a sus intereses convenga.

Lic. Pedro Geovanny Calva Jiménez
1150428496
Nro. Reg. Senecyt: 1031-2022-2421774

Abstract

The adaptation of pre-trained question-answering (QA) models is an essential task so that they can be implemented in different scenarios. The objective of this research is to obtain the value of the rough metric by applying the Fine-Tuning technique to the DistilBERT model to answer questions about the content extracted from academic tasks of the Computer Science Department of the National University of Loja. To develop this work, the CRISP-ML (Q) methodology was used as a reference framework, making use of its first four phases, in which the following was done: a compilation of 30 academic tasks obtained from 6 different subjects, from which 80 questions about their content were generated through crowdsourcing, which served as a basis for creating a dataset in SQuAD1.0 format with 1410 data, of which 800 were generated through paraphrasing and the Few-shot learning approach, and the remaining 610 with the direct contribution of the author. This dataset was divided into 90% for training (train) and 10% for evaluation (test), with an additional subdivision of the train set (75% for train and 25% for validation). Having the data prepared, DistilBERT hyperparameters were adjusted to train four different models using TensorFlow on the Google Colab platform with the GPU T4 runtime environment, selecting the best model based on its level of response extraction and F1-score. Once the QA model was chosen, an evaluation was performed using the ROUGE metric, including A/B testing. The QA model was deployed in Hugging Face and achieved an accuracy of 86.93% during its training with 51 epochs, a learning rate of $1e$ -5, and a batch size of 32, which through evaluation achieved a maximum F-measure in ROUGE-L of 60.96. These values demonstrate the importance of applying Fine-Tuning in the development of QA models for specific contexts.

Keywords: QA model, DistilBERT, SQuAD 1.0 dataset, CRISP-ML(Q), ROUGE