UNIVERSIDAD AUTÓNOMA DE AGUASCALIENTES

CENTRO DE CIENCIAS BÁSICAS

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

Tesina

“Comparación cualitativa y cuantitativa de respuestas generadas por Modelos de Lenguaje Pre Entrenados”

Presenta

Erick Alexis Galván Vargas

Para Obtener el Grado de Ingeniería en Computación Inteligente

Comité tutorial:

Dr. Alejandro Padilla Díaz

Dr. Francisco Javier Álvarez Rodríguez

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Aguascalientes, Ags., 20 de Junio de 2024

A quien corresponda:

Por medio de la presente me permito informar que el alumno(a) **Erick Alexis Galván Vargas** de la carrera de Ingeniería en Computación Inteligente con **ID: 270942**, ha terminado satisfactoriamente su tesina titulada: **”Comparación cualitativa y cuantitativa de respuestas de Modelos de Lenguaje Pre Entrenados”**, correspondiente a la materia de Seminario de Investigación II.

Para los fines que al interesado convengan.

ATENTAMENTE

Dr. Alejandro Padilla Díaz Director

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Aguascalientes, Ags., 20 de Junio de 2024

A quien corresponda:

Por medio de la presente me permito informar que el alumno(a) **Erick Alexis Galván Vargas** de la carrera de Ingeniería en Computación Inteligente con **ID: 270942**, ha terminado satisfactoriamente su tesina titulada: **"Comparación cualitativa y cuantitativa de respuestas de Modelos de Lenguaje Pre Entrenados"**, correspondiente a la materia de Seminario de Investigación II.

Para los fines que al interesado convengan.

ATENTAMENTE

Dr. Francisco Javier Álvarez Rodríguez
Miembro del Comité Tutoral.

Resúmen

La tecnología crece a pasos agigantados. Se ha escuchado esa frase tantas veces que se suele olvidar todo lo que implica. Cada día, los usuarios son más exigentes con los dispositivos que los rodean. “¿Por qué esta app tarda 2.004 segundos en iniciar?”, “Mi laptop se pone lenta cuando tengo 17 programas abiertos, ¡qué horror!”, “Spotify no me recomienda canciones de acuerdo a cómo me siento hoy, es horrible”.

Estas expresiones reflejan sentimientos reales de usuarios que lidian con la tecnología a cada momento. Hoy en día, se tiene el compromiso de mejorar continuamente para que los usuarios utilicen la tecnología de la manera más cómoda posible, reconociendo que siempre hay áreas de mejora.

En la actualidad, el área tecnológica que es tema de conversación a nivel mundial, tanto entre profesionales de las Tecnologías de la Información como entre el público en general, es la Inteligencia Artificial. No es algo completamente nuevo, ya que se lleva décadas trabajando con ella. Sin embargo, desde hace aproximadamente un año, la IA se ha convertido en una herramienta de uso cotidiano para prácticamente cualquier persona que tenga un smartphone.

La popularidad y el rápido crecimiento de la IA han impulsado un esfuerzo constante por mejorar esta tecnología, haciéndola más eficiente, rápida, precisa y, ¿más humana?

Esta tesina se enfocó en identificar las características que hacen que un modelo de lenguaje sea superior a otros. El objetivo fue comprender las fortalezas y debilidades de cada modelo, y obtener una idea más clara de qué y cómo replicar las mejores características en futuras construcciones de modelos.

Las pruebas se realizaron mediante una interfaz creada específicamente para este propósito. Esta interfaz permite subir un archivo PDF, del cual extrae la información y posibilita una conversación basada en su contenido, utilizando alguno de los tres modelos de lenguaje previamente seleccionados. El usuario puede ver las métricas cuantitativas de las respuestas obtenidas y calificar manualmente su percepción de la calidad de estas respuestas.

Los resultados obtenidos muestran los puntos fuertes de cada uno de los modelos, después de haber analizado las pruebas realizadas y dándole una interpretación a los valores obtenidos. Con ello, se pueden analizar las cualidades de cada modelo, notando las características que los hacen diferenciarse de los demás.

1. Introducción..... 6

2. Planteamiento del problema..... 7
3. Justificación..... 8
4. Objetivos..... 8
4.1 Objetivo general..... 9
4.2 Objetivos específicos..... 9
5 Preguntas de investigación..... 9
6 Fundamentos teóricos..... 9
6.1 La Inteligencia..... 10
6.2 La Inteligencia Artificial..... 11
6.3 Redes neuronales..... 12
6.4 Machine Learning..... 14
6.5 Procesamiento de Lenguaje Natural (PLN)..... 15
6.6 Modelos de Inteligencia Artificial..... 16
6.7 Modelos de lenguaje..... 16
6.8 Arquitecturas de modelos de lenguaje pre entrenados..... 17
6.9 Large Language Models..... 19
6.10 Arquitectura de Transformers..... 19
6.10.1 T5 (Text-to-Text Transfer Transformer)..... 20
6.11 Entrenamiento de LLMs..... 21
6.12 LangChain..... 21
6.13 Chunks..... 22
6.14 Embeddings..... 24
6.15 N-gramas..... 26
6.16 Métricas de evaluación de modelos..... 26
6.16.1 BERTScore..... 27
6.16.2 BLEU (Bilingual Evaluation Understudy)..... 28
6.16.3 ROUGE (Recall-Oriented Understudy for Gisting Evaluation)..... 29
6.16.6 Wiki Split..... 32
7. Metodología..... 34
8. Desarrollo..... 34
8.1 Modelos utilizados..... 35
8.1.1 OpenAI GPT 3.5 Turbo..... 35
8.1.2 Google Flan T5 Base..... 36
8.1.2 MistralAI Mistral-7B-Instruct-v0.2..... 38
8.2 Backend..... 39
8.2.1 Creación de server en FastAPi..... 39
8.2.2 HuggingFace..... 40
8.2.3 Subir un PDF..... 42
8.2.4 Realizar una pregunta..... 43
8.2.5 Testear el modelo con métricas computacionales..... 48
8.3 Frontend..... 50
8.3.1 Subir el archivo PDF..... 50
8.3.2 Hacer preguntas..... 52
8.3.3 Testear las respuestas..... 54

8.4 Métricas cualitativas.....60
8.4.1 Coherencia.....60
8.4.2 Relevancia.....61
8.4.3 Fluidez.....62
8.4.4 Creatividad.....62
9 Experimentación y pruebas.....64
9.1 OpenAI Gpt 3.5 Turbo.....64
9.1.1 Resultados cuantitativos.....64
9.1.2 Resultados cualitativos.....66
9.2 Google Flan T5 Base.....66
9.2.1 Resultados cuantitativos.....66
9.2.2 Resultados cualitativos.....68
9.3 MistralAI Mistral-7B-Instruct-v0.2.....68
9.3.1 Resultados cuantitativos.....68
9.3.2 Resultados cualitativos.....69
9.4 Gráficas de los resultados.....70
9.4.1 GPT 3.5 Turbo.....70
9.4.2 Flan T5 Base.....73
9.4.3 Mistral 7B Instruct.....76
10 Análisis e interpretación de resultados.....79
10.1 GPT 3.5 Turbo.....80
10.1.1 Precisión y coincidencia de palabras (BLEU y SacreBLEU).....81
10.1.2 ROUGE.....82
10.1.3 BERTScore.....83
10.1.4 WikiSplit.....83
10.1.5 Evaluación cualitativa.....84
10.1.6 Conclusiones.....84
10.2 Google Flan T5 Base.....85
10.2.1 BLEU.....85
10.2.2 ROUGE.....86
10.2.3 BERTScore.....86
10.2.4 WikiSplit.....87
10.2.5 Evaluación cualitativa.....87
10.3.6 Conclusiones.....88
10.3 MistralAI Mistral-7B-Instruct-v0.2.....88
10.3.1 BLEU.....88
10.3.2 ROUGE.....89
10.3.3 BERTScore.....89
10.3.4 WikiSplit.....90
10.3.5 Evaluación cualitativa.....90
10.3.6 Conclusiones.....91
10.4 Análisis general.....91
11 Conclusiones.....95
12 Referencias.....95

1. Introducción

En el siglo XXI, y especialmente en el año 2024, se ha alcanzado un punto de casi completa digitalización a nivel mundial. Los días en los que las oficinas administrativas estaban repletas de estantes con cajas polvorientas llenas de folders color paja, y documentos en los que se tenía que buscar durante horas para encontrar un dato específico, han quedado atrás.

Actualmente, para buscar el mismo dato, basta con abrir una computadora, teclear el nombre del archivo en el buscador y obtener la información deseada casi al instante. ¿Qué método preferiría una persona encargada de una notaría pública que haya trabajado desde 1990 hasta la actualidad y, por ende, conozca ambos métodos? Aunque para algunos la respuesta pueda parecer obvia, la verdad es que es difícil saberlo sin preguntarles directamente.

Es cierto que a veces resulta complicado adaptarse a los constantes cambios tecnológicos, especialmente para aquellas personas que no crecieron con la tecnología. No obstante, también es cierto que esta se ha convertido en una herramienta que facilita muchas tareas y, en este caso en concreto, reduce considerablemente el tiempo necesario para encontrar la información deseada, un factor cada vez más importante.

Con el avance tecnológico, que ha crecido a pasos agigantados, surge el tema de la inteligencia artificial. Aunque la IA no es algo nuevo y ha estado en desarrollo durante décadas, en tiempos recientes, con la llegada de generadores de imágenes, texto, voz, música y muchas más aplicaciones, se ha popularizado entre la gente, siguiendo los mismos pasos que en su momento tomaron dispositivos como los teléfonos inteligentes, hasta convertirse en una parte indispensable de la vida diaria moderna.

Independientemente de las polémicas y los innumerables temas de discusión que podrían surgir en torno a la inteligencia artificial, la realidad es que, al igual que las computadoras, el internet y la digitalización, la IA ha llegado como una herramienta más para ayudar en las actividades cotidianas.

Además de su impacto evidente en la eficiencia y accesibilidad, la inteligencia artificial también plantea cuestiones fundamentales sobre la privacidad, la seguridad y la ética. Estos desafíos, junto con la constante evolución de la tecnología, nos instan a reflexionar sobre cómo utilizamos estas herramientas poderosas y cómo podemos garantizar que beneficien a la sociedad en su conjunto.

Este proyecto se centra en la inteligencia artificial y la información, buscando comprender en profundidad qué hace que un modelo de lenguaje sea más poderoso que otros, especialmente en el contexto de su reciente crecimiento. Se investigará e intentará identificar qué aspectos necesitan reforzarse y replicarse al crear un modelo que genere texto. La investigación será principalmente

práctica, ya que se busca obtener la perspectiva de un usuario común, quien actualmente utiliza estas herramientas de manera cotidiana y a quien están dirigidos estos modelos.

¿Cómo se puede hacer que un usuario que sabe poco o nada sobre cómo está hecho un modelo lo evalúe? La respuesta es sencilla: haciendo que lo use. Se creará una interfaz web en la que se pondrán a prueba tres modelos de lenguaje seleccionados. El usuario podrá subir un archivo PDF, el sistema leerá la información y generará respuestas a las preguntas formuladas. Con una rúbrica, el usuario podrá puntuar la calidad de las respuestas recibidas, mientras que se utilizarán métricas computacionales existentes para evaluar estos modelos. Esta interfaz será la principal herramienta para alcanzar el objetivo: conocer y definir claramente las características esenciales que debe tener un modelo de lenguaje actual.

Un modelo de esta clase busca generar textos que se asemejen lo más posible a los producidos por un humano, hasta el punto en que quien lo pruebe no pueda distinguir si el ente con el que interactúa es humano o no. Por eso, las métricas cualitativas darán una perspectiva completamente humana y real al utilizar un modelo de manera cotidiana.

Las métricas cuantitativas ayudarán a medir la precisión de cada modelo, utilizando algoritmos matemáticos. Es importante conocer la exactitud del modelo a través de datos numéricos y palpables.

Ambas formas de medición son igualmente importantes y se les dará el mismo peso para formar la conclusión final. Será necesario tener una visión flexible para detectar diferentes escenarios y situaciones durante las pruebas, con el fin de obtener los mejores resultados.

2. Planteamiento del problema

En la actualidad, es muy común conocer grandes inteligencias artificiales como Chat GPT de OpenAI o Gemini de Google. Sabemos cómo usarlas y, a menudo, se aprovechan al máximo en tareas cotidianas.

Sin embargo, aunque estas tecnologías están respaldadas por empresas gigantes, ¿quiénes son las personas que las mantienen o que estuvieron involucradas en su construcción? Estas personas son ingenieros que probablemente aprendieron a programar siguiendo tutoriales en YouTube o leyendo artículos. En otras palabras, la mayoría de los ingenieros de hoy en día son autodidactas y su mayor fuente de aprendizaje es internet. Es probable que muchos de los ingenieros que actualmente trabajan en grandes empresas de inteligencia artificial, como las mencionadas anteriormente, crearan su primera red neuronal siguiendo estos recursos.

La información disponible en ese entonces no es la misma que existe hoy. Se ha avanzado mucho en los últimos años, en gran medida gracias a los modelos que estas personas han creado.

Por ello, es importante que se aproveche la información actual y se aprenda de los modelos ya desarrollados. Se deben utilizar sus aciertos como base de conocimiento para los futuros ingenieros en inteligencia artificial, o para cualquiera que desee entender cómo funcionan los modelos en la actualidad.

3. Justificación

Como se mencionó anteriormente, en el contexto actual abundan las inteligencias artificiales generadoras de texto. Algunas son más conocidas que otras y algunas son mejores que otras. Se puede discutir sobre qué hace que una IA sea mejor o más utilizada, mencionando factores como el marketing, los recursos empleados para crearla, el dinero invertido, la información con la que fue entrenada, entre otros. Incluso la empresa, el grupo o las personas que la crearon y las condiciones en las que lo hicieron tienen un papel fundamental. Pero, hablando de aspectos más técnicos, ¿qué se puede rescatar del proceso de construcción de estas IAs que se deban tener en cuenta al momento de crear una?

Es fundamental comparar los modelos existentes para tener una idea más clara de cómo se puede mejorar continuamente el desarrollo de los mismos a corto plazo. Los modelos de lenguaje están evolucionando día a día, y se necesita tomar lo mejor de cada uno para usarlo como referencia en la creación de nuevos modelos.

El problema central de este proyecto es identificar las características esenciales que definen un modelo de lenguaje de alta calidad desde la perspectiva de un usuario común, así como desde métricas computacionales.

El objetivo final es obtener una comprensión profunda de las características que deben reforzarse y replicarse en la creación de modelos de lenguaje, garantizando que sean eficaces y útiles para los usuarios finales. Además, se busca equilibrar las evaluaciones humanas y computacionales para ofrecer una visión integral del desempeño de estos modelos.

Este enfoque permitirá no solo mejorar la calidad de los modelos de lenguaje actuales, sino también establecer una base sólida para el desarrollo de futuras generaciones de IA, asegurando que estas herramientas continúen evolucionando de manera que maximicen su utilidad y aceptación por parte de los usuarios.

4. Objetivos

4.1 Objetivo general

Conocer qué hace que un modelo sea superior o inferior a otros modelos existentes y conocidos en la actualidad, basándose en la evaluación cuantitativa y cualitativa de sus respuestas generadas.

4.2 Objetivos específicos

- Seleccionar tres modelos de lenguaje capaces de generar texto a partir de una entrada proporcionada por el usuario.
- Crear una interfaz sencilla de usar que permita cargar un archivo PDF, para que el modelo aprenda la información contenida y pueda responder preguntas específicas sobre ese tema.
- Emplear métricas cualitativas para que diferentes usuarios califiquen la calidad de las respuestas en conversaciones con cada modelo.
- Utilizar métricas cuantitativas existentes para medir la precisión de las respuestas en una conversación típica con cada modelo.
- Recopilar los datos de las pruebas realizadas y, con base en estos resultados, concluir qué características hacen que un modelo sea superior a otros.

5 Preguntas de investigación

Para poder crear un proyecto que se ajuste al objetivo que se tiene en mente, se debe saber cúal es el mejor camino para llegar a él. Por eso se consideran las siguientes preguntas de investigación.

- ¿Qué modelos serán los adecuados para realizar las pruebas?
- ¿Qué métricas serán mejores tanto cualitativa como cuantitativamente para probar los modelos?
- ¿Qué tecnologías utilizar para realizar la interfaz en la que se van a hacer las pruebas?
- ¿De qué manera se pueden interpretar los datos obtenidos para tener la mejor conclusión posible?

6 Fundamentos teóricos

Partiendo de conceptos previamente mencionados, como la inteligencia artificial y la necesidad de optimizar la búsqueda de información en documentos digitales, el proyecto propuesto es el siguiente: Recopilar información sobre el rendimiento de modelos de generación de texto existentes a partir de sus respuestas generadas para así saber cuál y por qué es el mejor. Esto será posible con la ayuda de una interfaz que se explica a continuación:

Una página web donde el usuario pueda cargar documentos de texto. A partir de la información contenida en el documento, el sistema deberá aprender de él y, de esta manera, ser capaz de responder preguntas que realice el usuario a través de un chat.

La página web debe ser lo más sencilla e intuitiva posible. La pantalla estará dividida en dos: la mitad izquierda tendrá un botón para cargar el documento de texto si aún no se ha hecho; si ya se cargó, se mostrará una vista previa del mismo. En la otra mitad estará el chat, con todos los mensajes tanto del usuario como del sistema sobre la información proporcionada.

Una vez que el documento se haya cargado, su información debe ser leída, limpiada y procesada mediante Procesamiento de Lenguaje Natural (PLN), permitiendo así que un modelo de inteligencia artificial pueda aprender de él.

Para llevar a cabo este proyecto de manera óptima y cumplir con todos los objetivos, es fundamental comprender breves conceptos teóricos que se presentarán a continuación:

6.1 La Inteligencia

Desde los primeros grupos de humanos recolectores hasta grandes civilizaciones como la griega, ha existido un profundo deseo e interés por comprender qué distingue al ser humano de otras formas de vida en la Tierra.

La capacidad de pensar y razonar se remonta a tiempos prehistóricos, cuando los antepasados del homosapiens más lejanos comenzaron a utilizar herramientas para cazar, desarrollaron formas de comunicación cada vez más complejas y experimentaron cambios en el tamaño y la distribución del cerebro. (Asale, R.-. (s. f.))

En el siglo XX, los psicólogos iniciaron el desarrollo de pruebas estandarizadas para medir la inteligencia. Alfred Binet, pionero en este campo, creó el primer test de inteligencia para evaluar las habilidades cognitivas de los niños. A partir de entonces, se desarrollaron diversas pruebas, como el famoso coeficiente intelectual (CI), con el objetivo de cuantificar la inteligencia de manera objetiva. (Biografia de Alfred Binet. (s. f.))

Con el tiempo, los enfoques cognitivos ganaron relevancia. La teoría de las inteligencias múltiples de Howard Gardner propuso que la inteligencia no se limita a un solo factor, sino que abarca diversas habilidades, como la lingüística, la lógico-matemática, la musical, entre otras. Esta perspectiva ampliada resaltó la diversidad de las capacidades cognitivas humanas.

Según la Real Academia Española (RAE), (Rae, & Rae. (n.d.)) el concepto de inteligencia se define como:

“1. f. Capacidad de entender o razonar.”

“3. f. Comprensión de algo”

Rae, & Rae. (n.d.)

Un concepto general de inteligencia podría ser la “capacidad de resolver problemas basándose en la comprensión de los mismos”.

Como se mencionó anteriormente, existen muchos tipos de inteligencia. Incluso se puede observar que la inteligencia no es exclusiva del ser humano. Muchos animales utilizan su propia inteligencia en cuestiones de supervivencia, alimentación e incluso recreación.

¿Qué diferencia a la inteligencia humana de otros tipos de inteligencia? ¿Se puede ser capaz de distinguir adecuadamente la inteligencia humana de la de otros seres?

6.2 La Inteligencia Artificial

Actualmente, prácticamente todos conocen y han tenido en sus manos algún dispositivo como una computadora o celular, que a su vez realiza, por medio de alguna aplicación, alguna tarea que los ha dejado boquiabiertos. Pero, ¿este tipo de programas son realmente inteligentes? Aunque muchas de esas aplicaciones están programadas con cierto algoritmo que pueda hacer creer eso (cada vez menos frecuentemente), al final de cuentas es eso, pura programación completamente preparada para realizar acciones ya definidas.

Las necesidades de explorar e ir más allá en la computación nos hacen llegar al tema de la inteligencia artificial. Se puede definir la inteligencia artificial como la capacidad de las máquinas para realizar tareas que normalmente requieren inteligencia humana. Esto incluye el aprendizaje, la adaptación, la resolución de problemas y la interacción con el entorno.

La noción de crear máquinas capaces de imitar la inteligencia humana tiene raíces profundas. En la antigüedad, mitos y leyendas mencionan autómatas y seres artificiales. Sin embargo, la formalización de la idea de máquinas inteligentes comenzó en el siglo XX. En la década de 1950, Alan Turing propuso el famoso "Test de Turing", (Nodd3r. (n.d.).) un experimento mental que plantea la pregunta: ¿puede una máquina exhibir un comportamiento indistinguible del de un ser humano? Este concepto sentó las bases teóricas para la inteligencia artificial (IA). (Walther. (2023, 2 mayo))

Antes de abordar la IA, Turing desarrolló la idea de una "máquina universal", un concepto clave en la teoría de la computación. Esta máquina teórica podría realizar cualquier cálculo que pudiera describirse de manera algorítmica. Esta conceptualización sentó las bases para la arquitectura de las computadoras modernas. En sus escritos, Turing exploró el concepto de una máquina abstracta (ahora conocida como la Máquina de Turing) que podía realizar cualquier tarea computacional.

También introdujo el "Juego del Imitador", precursor del Test de Turing, donde una máquina trata de imitar a un humano en una conversación escrita.

Los primeros años de la IA se centraron en tareas específicas, como el ajedrez y el procesamiento de información simbólica. El programa de ajedrez desarrollado por Claude Shannon en 1950 y el "Logic Theorist" de Allen Newell y Herbert A. Simon en 1955 fueron hitos tempranos; y un par de décadas más tarde, el 11 de mayo de 1997, se produciría la primera victoria de una máquina contra un humano: la supercomputadora Deep Blue desarrollada por IBM derrotaba al Gran Maestro Garry Kasparov. (Prego, C. (2022, 20 febrero))

6.3 Redes neuronales

Las redes neuronales, inspiradas en el funcionamiento del cerebro humano, son un componente fundamental en el campo de la inteligencia artificial (IA). El concepto de una red neuronal artificial se remonta a los años 40 y 50 del siglo pasado, cuando Warren McCulloch y Walter Pitts desarrollaron un modelo simplificado de neurona, la unidad básica de procesamiento en una red neuronal. Propusieron que estas unidades podrían organizarse para realizar funciones lógicas, sentando las bases teóricas para las redes neuronales, diseñadas para realizar tareas específicas al aprender patrones y relaciones complejas a partir de datos. (¿Qué son las redes neuronales? | IBM. (s. f.))

Una red neuronal consta de ciertos componentes básicos: (*SPSS Modeler Subscription*. (2021, August 17))

- Neurona Artificial: Es la unidad básica de una red neuronal. Recibe entradas ponderadas, las suma y aplica una función de activación para producir una salida.
- Conexiones y Pesos: Las conexiones entre neuronas están representadas por pesos, que indican la importancia de la entrada. Estos pesos se ajustan durante el entrenamiento para optimizar el rendimiento de la red.
- Capas: Las redes neuronales se organizan en capas. La capa de entrada recibe datos, las capas ocultas procesan la información, y la capa de salida produce la respuesta final.

El funcionamiento básico de una red neuronal sigue el siguiente proceso:

1. Entrada: La información o datos de entrada se proporciona a la capa de entrada, y cada neurona recibe una parte de estos datos ponderados por los pesos.
2. Procesamiento: En cada neurona, la suma ponderada de las entradas se pasa a través de una función de activación. Esta función determina si la neurona debe activarse y cuál será su salida.

3. Propagación hacia Adelante: La salida de una capa se convierte en la entrada para la siguiente capa. Este proceso se repite hasta llegar a la capa de salida.
4. Comparación con la Salida Deseada: La salida predicha se compara con la salida deseada (etiqueta real) utilizando una función de pérdida, que mide la discrepancia entre la predicción y la verdad conocida.
5. Retropropagación: Se utiliza el algoritmo de retropropagación para ajustar los pesos de las conexiones en la red, minimizando la función de pérdida. Este proceso se repite iterativamente durante el entrenamiento.
6. Entrenamiento: La red "aprende" ajustando los pesos para mejorar la precisión de las predicciones, repitiendo este proceso hasta alcanzar un rendimiento deseado.

Con el tiempo, se han creado diferentes tipos de redes neuronales, cada una con un propósito y funcionamiento acorde a sus características específicas:

- ➔ Perceptrón: La unidad básica con una sola capa. Recibe entradas ponderadas, las suma y aplica una función de activación para producir una salida. Útil para problemas lineales simples y tareas de clasificación lineal.

- ➔ Redes Neuronales Multicapa (MLP): Con capas ocultas entre la entrada y la salida. Las capas ocultas permiten aprender patrones más complejos. Cada neurona en una capa oculta procesa información y la pasa a la siguiente capa. Ampliamente utilizado para problemas de clasificación y regresión más complejos.

- ➔ Redes Neuronales Convolucionales (CNN): Eficientes para datos de tipo imagen. Utilizan filtros convolucionales para identificar características en regiones específicas de la imagen. Útiles para reconocimiento de objetos en imágenes. Ampliamente utilizado en visión por computadora.

- ➔ Redes Neuronales Recurrentes (RNN): Adecuadas para datos secuenciales. Cada paso de tiempo procesa la entrada actual y la información almacenada de pasos anteriores. Útiles para tareas que dependen del contexto temporal. Se utilizan en traducción automática, análisis de sentimientos y predicción de series temporales.

- ➔ Redes Generativas Adversarias (GAN): Comprenden un generador y un discriminador, utilizadas para generación de imágenes realistas y otras tareas creativas. El generador crea datos, y el discriminador intenta distinguir entre datos reales y generados. Ambos mejoran iterativamente en su tarea.

- ➔ Redes Neuronales Transformer: Basadas en la atención y mecanismos de autoatención. Permiten procesar datos secuenciales en paralelo, mejorando la eficiencia en el

entrenamiento. Ampliamente utilizado en procesamiento de lenguaje natural. Ampliamente utilizado en procesamiento de lenguaje natural.

➔ Redes Neuronales Long Short-Term Memory (LSTM) y Gated Recurrent Unit (GRU): Variantes de RNN diseñadas para abordar problemas de desaparición del gradiente. Introducen mecanismos de puertas para controlar el flujo de información. Conservan la memoria a largo plazo y a corto plazo. Se usan en tareas con dependencias a largo plazo.

Las redes neuronales se aplican en diversas áreas, como reconocimiento de imágenes, procesamiento de lenguaje natural y recomendación de productos. Simulan el proceso de aprendizaje y adaptación observado en el cerebro humano, permitiendo a las máquinas realizar tareas complejas mediante la extracción y comprensión de patrones en datos.

6.4 Machine Learning

El Machine Learning, o aprendizaje automático, es una rama de la inteligencia artificial que se enfoca en desarrollar algoritmos y modelos que permitan a las máquinas aprender patrones a partir de datos y mejorar su rendimiento sin intervención humana directa. El proceso de aprendizaje implica la exposición del sistema a datos, la identificación de patrones y la adaptación del modelo para hacer predicciones o tomar decisiones. (¿Qué es machine learning? | IBM. (s. f.))

- Supervisado vs. No Supervisado:
  Supervisado: El modelo se entrena con un conjunto de datos que incluye ejemplos etiquetados, es decir, datos emparejados con las respuestas correctas. El objetivo es que el modelo aprenda a hacer predicciones o clasificaciones basadas en estos ejemplos.
  No Supervisado: El modelo se entrena con datos que no están etiquetados. El sistema debe identificar patrones o estructuras en los datos por sí mismo, como agrupamiento o reducción de dimensionalidad.

- Aprendizaje Profundo (Deep Learning):
  Usa redes neuronales profundas para poder aprender representaciones complejas de datos. Es especialmente eficaz en tareas como reconocimiento de imágenes, procesamiento de lenguaje natural y juegos estratégicos.

- Reforzado (Reinforcement Learning):
  Implica que un agente interactúe con un entorno y aprenda a tomar decisiones para maximizar alguna noción de "recompensa". Es comúnmente utilizado en juegos, robótica y toma de decisiones autónoma.

6.5 Procesamiento de Lenguaje Natural (PLN)

La forma en que la gente se comunica con las computadoras es mediante lenguajes de programación. El código se interpreta o compila según el caso, y después se lleva a cabo una serie de procesos que, al final, proporcionarán las instrucciones esperadas a la máquina. El Procesamiento de Lenguaje Natural es un campo multidisciplinario que combina la lingüística computacional, la inteligencia artificial y la lingüística teórica. Se enfoca en la interacción entre las computadoras y el lenguaje humano, de manera que se puedan dar instrucciones a las máquinas con lenguaje humano, y la máquina sea capaz de interpretarlas.

Referente al PLN, hay algunos conceptos básicos que nos ayudan a entender la forma en la que se intenta conseguir esto:

- Tokenización:
  La tokenización es el proceso de dividir un texto en unidades más pequeñas llamadas "tokens". Estos tokens pueden ser palabras, frases o incluso caracteres, dependiendo del nivel de granularidad necesario.
  Es el primer paso crucial en el análisis de texto, ya que ayuda a convertir un flujo de palabras en unidades manejables, facilitando así el procesamiento ulterior.

- Análisis Morfológico:
  El análisis morfológico se refiere al estudio de la estructura y la formación de palabras en un idioma. Implica analizar las raíces, los afijos y otras características morfológicas de las palabras.
  Comprender la morfología de las palabras es esencial para interpretar su significado en contexto, lo que mejora la capacidad de la máquina para entender el lenguaje humano de manera más profunda.

- Lematización:
  La lematización es el proceso de reducir una palabra a su forma base o lema. Por ejemplo, "programando" se lematiza como "programar".
  La lematización ayuda a simplificar el análisis al reducir las palabras a sus formas fundamentales, lo que facilita la identificación de patrones y la comprensión del significado subyacente.

6.6 Modelos de Inteligencia Artificial

En términos generales, un modelo es una representación simplificada o abstracta de un sistema o fenómeno del mundo real. Los modelos son herramientas conceptuales o matemáticas que ayudan a comprender, explicar y prever el comportamiento de algo más complejo. Por ejemplo, un modelo de clasificación de imágenes para detectar perros o gatos simula la capacidad humana de diferenciar entre esos animales.

Algunos usos comunes que se le dan a los modelos son:

- Predicción: Prever cómo cambiará un sistema en el futuro según ciertos parámetros.
- Comprensión: Ayudar a entender la estructura y el funcionamiento de un sistema.
- Optimización: Encontrar la mejor solución posible para un problema dado.
- Simulación: Reproducir el comportamiento de un sistema en un entorno controlado.

Y los pasos que se siguen para la creación de un modelo desde cero son:

- Formulación: Definir el propósito y los objetivos del modelo.
- Estructuración: Diseñar la representación del sistema y las relaciones entre variables.
- Calibración: Ajustar el modelo para que se ajuste a los datos y al comportamiento observado.
- Validación: Verificar si el modelo produce resultados precisos y coherentes.
- Utilización: Aplicar el modelo para realizar predicciones o tomar decisiones.

Se pueden mencionar algunos ejemplos de modelos, como podría ser un modelo meteorológico, que ayuda a predecir el clima mediante datos atmosféricos. O un modelo económico, que podría evaluar el rendimiento económico de algún activo basándose en variables financieras.

Ahora que se tiene un entendimiento más general de lo que es un modelo, es hora de enfocarse en el tipo de modelo que en este caso serviría para el proyecto: un modelo de lenguaje.

6.7 Modelos de lenguaje

Un modelo de lenguaje es una construcción matemática que asigna probabilidades a secuencias de palabras. En otras palabras, es capaz de predecir la probabilidad de que una palabra o secuencia de palabras siga a otra en un contexto dado. Este enfoque se ha vuelto esencial para entender, generar y procesar el lenguaje humano.

Con el tiempo, los modelos de lenguaje han evolucionado. Desde las primeras técnicas basadas en reglas hasta los enfoques más modernos basados en redes neuronales.

Dentro de las arquitecturas de modelos de lenguaje, destacan las redes neuronales recurrentes (RNN), como LSTM y GRU, que capturan dependencias a largo plazo. Sin embargo, los modelos basados en Transformers han emergido como líderes, permitiendo una atención global y procesamiento paralelo.

Algunas aplicaciones que se les dan a los modelos de lenguaje son:

-
- Generación de texto: Crear texto coherente y relevante, desde chatbots hasta resúmenes automáticos.
- Traducción automática: Facilitar la traducción de idiomas al comprender y generar texto en diferentes lenguajes.
- Reconocimiento de voz: Transformar la voz en texto mediante la comprensión de patrones lingüísticos.
- Búsqueda de información: Mejorar la precisión de los motores de búsqueda al comprender las consultas de manera más contextual.
- Asistentes virtuales: Potenciar la interacción con máquinas al comprender y responder preguntas de manera más natural.

Estas son solo algunas de las aplicaciones más usuales, pero las posibilidades para utilizar modelos de lenguaje son tantas como la imaginación lo permita.

6.8 Arquitecturas de modelos de lenguaje pre entrenados

Crear un modelo desde 0 es un trabajo complicado, no solo en la implementación, si no, también en el aspecto de que se necesitan buenos recursos computacionales, cantidades de datos bastante grandes, y tiempo para entrenarlo.

Una alternativa que existe para crear un modelo, son los modelos pre entrenados.

Estos modelos son algoritmos de aprendizaje automático que se entrenan previamente en grandes cantidades de datos antes de ser afinados para tareas específicas. Algunos de los modelos más destacados incluyen GPT (Generative Pre-trained Transformer) y BERT (Bidirectional Encoder Representations from Transformers).

- GPT (Generative Pre-trained Transformer):

    - Arquitectura: Basado en la arquitectura Transformer, que utiliza mecanismos de atención para procesar y generar texto.
    - Pre Entrenamiento: Entrenado en enormes conjuntos de datos para aprender patrones de lenguaje y estructuras gramaticales.
    - Generación de Texto: Puede generar texto coherente y contextualmente relevante dado un contexto inicial.
    - Aplicaciones: Traducción automática, resumen de texto, generación de contenido.

- BERT (Bidirectional Encoder Representations from Transformers):
    - Característica Clave: Utiliza la bidireccionalidad en el procesamiento de texto, considerando el contexto tanto a la izquierda como a la derecha de una palabra.
    - Pre Entrenamiento: Expuesto a grandes cantidades de texto en dos direcciones, lo que mejora la comprensión contextual.
    - Ventajas: Capacidad para entender el significado de una palabra en función de su contexto completo.
    - Aplicaciones: Responder preguntas, análisis de sentimientos, completar oraciones.

- Transformadores en Lenguaje Natural:
    - Arquitectura: Basada en el mecanismo de atención de Transformer.
    - Pre Entrenamiento: Modelos entrenados en grandes cantidades de datos, a menudo en tareas de autocompletado.
    - Generalidad: Capacidad para aplicarse a una variedad de tareas de procesamiento de lenguaje natural.
    - Ejemplos: GPT, BERT, XLNet.

- XLNet:
    - Enfoque: Incorpora el concepto de permutación de palabras en lugar de predecir palabras en orden secuencial.
    - Ventajas: Mejora la relación de dependencia a largo plazo y la generalización del modelo.

- Uso en Transferencia de Aprendizaje:
    - Fine-Tuning: Después del pre entrenamiento, los modelos se afinan para tareas específicas con conjuntos de datos más pequeños y etiquetados.

Estos modelos pre entrenados han demostrado su eficacia al capturar patrones semánticos y contextuales en el lenguaje natural. La transferencia de aprendizaje, donde los modelos pre entrenados se adaptan a tareas específicas, ha simplificado el desarrollo de aplicaciones de procesamiento de lenguaje natural, llevando a avances significativos en la comprensión y generación de texto.

6.9 Large Language Models

Los modelos lingüísticos grandes (LLMS por sus siglas en inglés) son programas que reconocen y generan lenguaje natural, aprendiendo patrones y teniendo la capacidad de manejar grandes cantidades de texto. (Cloudflare. (n.d.))

Tales modelos se caracterizan por ser entrenados con cantidades enormes de información, así como contar con millones de parámetros. Gracias a eso, pueden comprender y generar texto sobre una gran cantidad de temas y contextos.

La mayoría de los LLMs modernos, como GPT-4, BERT y T5, se basan en la arquitectura de transformers, que utiliza mecanismos de atención para manejar relaciones contextuales a lo largo de secuencias de palabras.

Los modelos de lenguaje a gran escala destacan por su capacidad para responder de manera impredecible. A diferencia de los programas tradicionales que siguen una serie de instrucciones predefinidas, estos modelos pueden entender el lenguaje humano y responder preguntas sobre temas específicos.

Sin embargo, su precisión depende completamente de la calidad de la información con la que fueron entrenados. Si se les proporciona información incorrecta, sus respuestas también serán incorrectas. Asimismo, cuando se les pregunta sobre temas que desconocen, pueden llegar a "inventar" información.

6.10 Arquitectura de Transformers

Los LLMs utilizan un tipo específico de redes neuronales conocidas como modelos transformadores. Estos modelos son especialmente eficaces para comprender el contexto, un aspecto crucial del lenguaje humano. Los modelos transformadores emplean una técnica llamada autoatención, que les permite detectar relaciones sutiles entre los elementos de una secuencia. Esto les confiere una comprensión del contexto superior a la de otros enfoques de aprendizaje automático, permitiéndoles relacionar el final de una frase con su inicio y conectar frases dentro de un párrafo.
Esta arquitectura permite a los modelos procesar palabras en paralelo y capturar dependencias a largo plazo en el texto mediante mecanismos de autoatención. Esto contrasta con los modelos secuenciales más antiguos como los LSTM y GRU, que procesan palabras de manera secuencial y pueden tener dificultades para capturar contextos de largo alcance. (*BM*. (n.d.))

Estos son dos puntos muy importantes para el funcionamiento de los transformadores:

- Autoatención: Permite que el modelo preste atención a todas las palabras en una secuencia de entrada, capturando relaciones contextuales importantes entre palabras.

- Encoders y Decoders: Los transformers típicamente consisten en capas de encoders (que procesan la entrada) y decoders (que generan la salida), aunque algunos modelos, como BERT, utilizan solo la parte de encoder para tareas de comprensión del lenguaje. (Merritt, R. (2022, September 16))

Los mecanismos de atención funcionan de la siguiente manera:

- Atención auto-regresiva (self-attention): Permite que cada posición en la secuencia de entrada (o salida) considere todas las demás posiciones para calcular una representación ponderada.

- Atención cruzada (cross-attention): En el decodificador, permite que cada posición en la secuencia de salida considere todas las posiciones en la representación del codificador.

Cada capa del codificador y decodificador incluye una capa de atención multi-cabezal y una red neuronal feed-forward completamente conectada. Además, los transformers hacen uso de normalización por capas (layer normalization) y técnicas de regularización como Dropout para mejorar la estabilidad y rendimiento del entrenamiento.

6.10.1 T5 (Text-to-Text Transfer Transformer)

T5 (Text-to-Text Transfer Transformer) es un modelo de procesamiento de lenguaje natural (NLP) diseñado para abordar una amplia variedad de tareas de NLP utilizando un enfoque unificado de "texto a texto". Esto significa que tanto la entrada como la salida del modelo son texto, permitiendo que el mismo modelo se aplique a diferentes tareas reformulándolas como problemas de transformación de texto. La arquitectura T5 (Text-to-Text Transfer Transformer) y la arquitectura general de Transformers comparten muchas similitudes, ya que T5 está basado en la arquitectura de Transformers, pero también tienen diferencias clave en su diseño y enfoque.

T5 convierte todas las tareas de NLP en un formato de entrada-salida de texto a texto. Por ejemplo, la entrada podría ser "traducir inglés a francés: Hello world" y la salida sería "Bonjour le monde". Este enfoque permite una gran flexibilidad y consistencia en la aplicación del modelo a diferentes tareas. Utiliza tokenización basada en subpalabras (Byte Pair Encoding, BPE), lo que permite manejar eficientemente un vocabulario extenso y mejorar el procesamiento de palabras raras y morfología

compleja. Debido a su enfoque de texto a texto, T5 puede ser entrenado simultáneamente en múltiples tareas, lo que ayuda a mejorar su capacidad de generalización y desempeño en tareas específicas. T5 puede ser aplicado a una variedad de tareas de NLP, incluyendo traducción de idiomas, resumen de textos, clasificación de texto, respuesta a preguntas, generación de texto, entre otras.

Las diferencias principales que tiene con la arquitectura de transformers, son que T5 reformula todas las tareas como problemas de transformación de texto a texto, mientras que la arquitectura Transformer original puede ser utilizada en una variedad de configuraciones y tareas, no necesariamente como texto a texto. Además T5 se beneficia de un extenso pre-entrenamiento y luego se ajusta en tareas específicas, mientras que los Transformers originales pueden ser entrenados desde cero para tareas específicas o usar diferentes estrategias de pre-entrenamiento.

T5 está optimizado para ser un modelo versátil y generalista en el procesamiento de lenguaje natural, mientras que los Transformers originales son más genéricos y pueden requerir más personalización para tareas específicas.

6.11 Entrenamiento de LLMs

El entrenamiento de LLMs implica el uso de grandes conjuntos de datos y recursos computacionales significativos. El proceso generalmente incluye:

- Preentrenamiento: El modelo se entrena inicialmente en un vasto corpus de texto sin etiquetar utilizando tareas como el modelado de lenguaje (predecir la siguiente palabra en una secuencia) o el llenado de máscaras (predecir palabras ocultas en una oración).

- Ajuste Fino (Fine-Tuning): Después del preentrenamiento, el modelo se ajusta con conjuntos de datos más pequeños y específicos para tareas particulares, mejorando su rendimiento en aplicaciones concretas.

6.12 LangChain

LangChain es una herramienta para crear aplicaciones que utilicen LLMs, mejorando en gran medida la experiencia al utilizar dichos modelos, personalizando, contextualizando y mejorando la precisión de las respuestas generadas.

Los LLMs son muy poderosos para generar respuestas en general, pero no funcionan de la mejor manera cuando se trata de temas por los que no han sido entrenados. LangChain entra aquí ofreciendo un contexto claro al modelo.

La forma en la que funciona LangChain es:

1. LangChain utiliza técnicas avanzadas de análisis lingüístico para descomponer el texto en unidades semánticas más pequeñas, como palabras, frases y párrafos.
2. A través de técnicas de modelado de contexto, LangChain captura las relaciones y conexiones entre las diferentes partes del texto, permitiendo una comprensión más completa del significado.
3. LangChain incorpora conocimientos externos, como bases de datos o ontologías, para enriquecer la comprensión del texto y proporcionar información adicional sobre entidades o conceptos mencionados.
4. Al comprender el contexto del texto de entrada, LangChain es capaz de generar respuestas que son coherentes y relevantes en función de la consulta o pregunta realizada.

6.13 Chunks

Los chunks no son conceptos exclusivos de la inteligencia artificial, sino que están más relacionados con el lenguaje. A menudo, cuando alguien quiere aprender un nuevo idioma, puede resultar complicado memorizar tantas palabras o frases juntas que pueden existir. Por eso existen los chunks, que no son más que grupos de palabras que comúnmente se encuentran juntas y que son similares semánticamente. Esto ayuda a dividir textos que podrían ser extensos en agrupaciones más pequeñas y manejables. (Weller, D. (2022, November 5))

Las aplicaciones principales de los chunks son:

- Los chunks se forman al identificar grupos de palabras que están estrechamente relacionadas semánticamente y que funcionan juntas para transmitir una idea o concepto específico.

- Los chunks ayudan a organizar la estructura sintáctica de una oración, dividiéndola en segmentos más manejables y comprensibles para el procesamiento lingüístico.

- Al agrupar palabras relacionadas, los chunks facilitan la comprensión del significado global de una oración, permitiendo a los modelos de lenguaje capturar la información relevante de manera más eficiente.

- Los chunks tienen una variedad de aplicaciones en el procesamiento del lenguaje natural, incluyendo el análisis de sentimientos, la extracción de información y la generación de resúmenes de texto.

En LLMs actuales, los chunks son una parte muy importante para su funcionamiento, ayudando en los siguientes puntos:

1. División del Texto en Fragmentos Gestionables:
    - En el procesamiento del lenguaje natural, los chunks se utilizan para dividir el texto en fragmentos más pequeños y manejables. Esto permite que los modelos de lenguaje procesen grandes volúmenes de texto de manera más eficiente y con menor consumo de recursos computacionales.
    - Durante la fase de preprocesamiento, el texto se divide en chunks que pueden ser oraciones, frases o párrafos. Estos fragmentos se analizan individualmente y luego se combinan para formar una comprensión más completa del texto.

2. Análisis Sintáctico y Semántico:
    - El texto se etiqueta con categorías gramaticales, como sustantivos, verbos y adjetivos. Estas etiquetas ayudan a identificar chunks significativos que conforman las estructuras sintácticas básicas de la oración.
    - Los chunks se utilizan para identificar y clasificar entidades en el texto, como nombres de personas, lugares y organizaciones. Esto facilita la extracción de información relevante del texto.

3. Mejora de la Comprensión del Contexto:
    - Al agrupar palabras relacionadas semánticamente, los chunks permiten a los modelos de lenguaje capturar el contexto local de una oración. Esto mejora la comprensión del significado y las relaciones entre las palabras.
    - Los chunks ayudan a resolver ambigüedades en el lenguaje natural, proporcionando un contexto adicional que permite a los modelos desambiguar significados potenciales de palabras o frases. (Equipo editorial, Etecé. (2021, August 5))

4. Entrenamiento y Evaluación de Modelos:
    - Durante el entrenamiento de modelos de lenguaje, los conjuntos de datos se dividen en chunks para mejorar la eficiencia del entrenamiento y la evaluación. Esto permite a los modelos aprender de una manera más estructurada y organizada.
    - Los chunks permiten un ajuste fino más preciso de los modelos en tareas específicas, al proporcionar fragmentos de datos relevantes que el modelo necesita aprender y comprender.

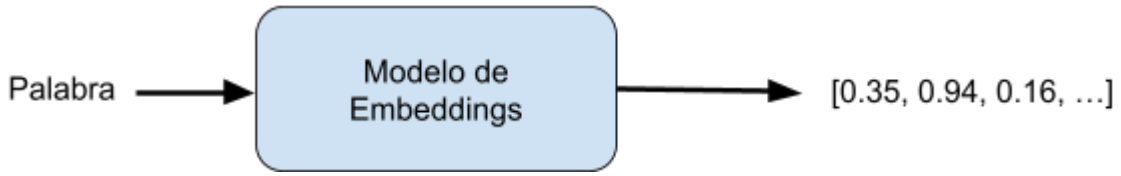
5. Generación de Texto:
    - Al generar texto, los modelos de lenguaje utilizan chunks para asegurar que las respuestas sean coherentes y fluidas. Los chunks permiten al modelo generar fragmentos de texto que se conectan de manera lógica y natural.
    - En la generación de resúmenes, los chunks ayudan a identificar las partes más importantes del texto original, permitiendo la creación de resúmenes concisos y representativos.

6.14 Embeddings

Cuando se quiere dar una instrucción a una computadora, como hacer clic en un botón para abrir una pestaña, el botón ejecuta una función escrita en un lenguaje de programación. Esta instrucción se convierte a un lenguaje de más bajo nivel, como el lenguaje ensamblador, y finalmente se traduce a lenguaje binario, que es lo que la máquina entiende. En resumen, las máquinas entienden lenguajes numéricos. (Espíndola, G. (2023, March 3))

Pero, si se desea dar instrucciones en lenguaje humano a una computadora, ¿cómo se puede hacer? ¿Cómo se puede convertir el lenguaje humano a una forma numérica?

Los embeddings son una técnica que convierte el lenguaje humano en una forma numérica para que una computadora pueda trabajar con él. Esta técnica convierte cada palabra en un vector numérico denso (en el que la mayoría de las entradas son diferentes de cero) de hasta 1000 dimensiones, donde cada dimensión captura una característica de la palabra en cuestión.



Galván, 2024, “Modelo de Embeddings”

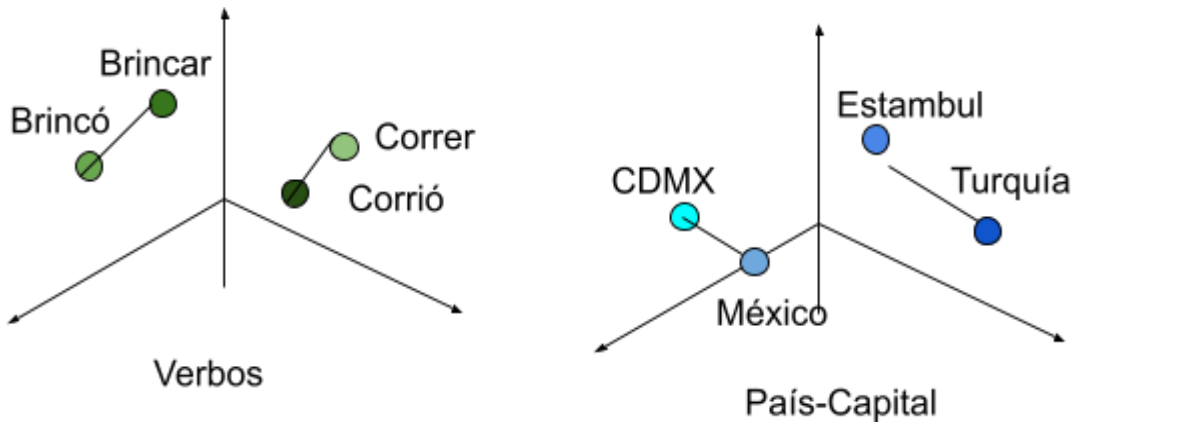
Las palabras similares semánticamente también tienen vectores similares. Por ejemplo, “pastel” y “postre” son palabras que se usan en contextos y situaciones parecidas, por lo que tendrían vectores similares. En contraste, palabras como “té” y “te”, aunque se escriben casi igual, tienen significados completamente diferentes, por lo que sus vectores serían distintos.

Los embeddings se obtienen a partir de modelos de aprendizaje automático entrenados con grandes corpus de texto. Durante este entrenamiento, los modelos aprenden las relaciones semánticas entre

las palabras analizando los contextos en los que aparecen. Ajustan los pesos de la red neuronal para representar palabras similares de manera cercana en el espacio vectorial.

Las ventajas y razones por las cuales la mayoría de los LLMs actuales utilizan embeddings son las siguientes:

- Los embeddings permiten a los modelos capturar relaciones semánticas complejas, como sinónimos y antónimos.
- Al considerar el contexto en el que aparecen las palabras, los embeddings pueden desambiguar palabras con múltiples significados y entender su uso en diferentes contextos.
- Al representar palabras en un espacio vectorial de baja dimensión, los embeddings reducen la complejidad computacional y mejoran la eficiencia en el procesamiento de grandes volúmenes de texto.
- Los embeddings pre entrenados pueden ser reutilizados en diferentes tareas de PLN, permitiendo la transferencia de aprendizaje y reduciendo la necesidad de entrenar modelos desde cero.
- Los embeddings mejoran significativamente el rendimiento de los modelos de lenguaje al proporcionar una representación rica y contextual del significado de las palabras.
- En la generación de texto, los embeddings permiten a los modelos producir respuestas coherentes y contextualmente apropiadas, mejorando la calidad de las interacciones con los usuarios.



Galván, 2024, “Palabras en el espacio vectorial”.

Como se puede ver, en el gráfico se hace una representación de cómo estarían distribuidos los embeddings de un texto. Por ejemplo, los verbos estarían en el mismo espacio vectorial, estando espacialmente cerca los mismos verbos pero que cuenten con distinto tiempo. En otro espacio vectorial estarían los países, teniendo cerca a sus capitales.

6.15 N-gramas

Un n-grama es una secuencia contigua de “n” elementos de un texto o de una cadena de habla. Los elementos pueden ser letras, sílabas, palabras, fonemas o incluso caracteres, dependiendo del tipo de análisis lingüístico o procesamiento del lenguaje natural (PLN) que se esté realizando. (Foqum Analytics, Empowers your success. (2023, November 17)

Tipos de N-gramas

- Unigrama (1-grama):
    - Consiste en elementos individuales.
    - Ejemplo: "El", "gato", "negro" (para palabras) o "E", "l", "g" (para caracteres).
- Bigramas (2-gramas):
    - Consiste en pares de elementos contiguos.
    - Ejemplo: "El gato", "gato negro" (para palabras) o "El", "lg", "ga" (para caracteres).
- Trigramas (3-gramas):
    - Consiste en tríos de elementos contiguos.
    - Ejemplo: "El gato negro", "gato negro duerme" (para palabras) o "Elg", "lga", "gat" (para caracteres).
- N-gramas de Mayor Longitud:
    - Secuencias de “n” elementos contiguos.
    - Ejemplo con n=4 para palabras: "El gato negro duerme".

Se considera el texto "El gato negro duerme plácidamente."

- Unigrama (1-grama):
    - ["El", "gato", "negro", "duerme", "plácidamente"]
- Bigramas (2-gramas):
    - ["El gato", "gato negro", "negro duerme", "duerme plácidamente"]
- Trigramas (3-gramas):
    - ["El gato negro", "gato negro duerme", "negro duerme plácidamente"]
- Cuatrigramas (4-gramas):
    - ["El gato negro duerme", "gato negro duerme plácidamente"]

6.16 Métricas de evaluación de modelos

Existe una serie de métricas utilizadas para medir la calidad y precisión de los modelos para garantizar que las respuestas generadas sean coherentes, precisas y relevantes, pero de manera

matemática y sin intervención humana directa. Estas son algunas de las métricas más utilizadas para este fin:

6.16.1 BERTScore

BERTScore es una métrica basada en el modelo de lenguaje BERT (Bidirectional Encoder Representations from Transformers). Evalúa la calidad de las respuestas generadas comparando embeddings contextuales de las palabras en las respuestas generadas y de referencia.

Funcionamiento:

- Utiliza BERT para generar embeddings de alta dimensión para cada palabra en la respuesta generada y en la respuesta de referencia.
- Los embeddings capturan información contextual, es decir, el significado de una palabra en relación con su contexto en la oración.
- Calcula la similitud entre cada par de palabras de las respuestas generada y de referencia utilizando una medida de similitud de coseno.
- Encuentra el mejor emparejamiento de palabras entre las dos respuestas, maximizando la similitud total.
- La puntuación final de BERTScore es la media de las similitudes de todos los emparejamientos de palabras.
- Se pueden calcular tres variantes de BERTScore: Precision, Recall y F1 Score:

1. Precision
La precisión en BERTScore mide qué tan bien los tokens del texto generado coinciden con los tokens del texto de referencia en términos de sus embeddings.

Para cada token en el texto generado, encuentra el token más similar en el texto de referencia (utilizando la distancia coseno entre los embeddings).Luego calcula el promedio de las similitudes máximas encontradas.

Alta precisión indica que la mayoría de los tokens del texto generado tienen tokens similares en el texto de referencia.
2. Recall
El recall en BERTScore mide qué tan bien los tokens del texto de referencia coinciden con los tokens del texto generado en términos de sus embeddings.

Para cada token en el texto de referencia, encuentra el token más similar en el texto generado.

Alto recall indica que la mayoría de los tokens del texto de referencia tienen tokens similares en el texto generado.

3. F1 Score

El F1 Score en BERTScore es la media armónica de la precisión y el recall. Proporciona un balance entre estos dos valores.

$$F1\ Score\ =\ 2\ \bullet\ \frac{Precision \bullet Recall}{Precision + Recall}$$

Un alto F1 Score indica que el texto generado no solo es preciso sino también completo en relación con el texto de referencia. También proporciona una medida equilibrada de la calidad del texto generado.

Ventajas:

- Captura mejor el contexto y el significado que las métricas tradicionales basadas en n-gramas.
- Más robusta ante sinónimos y variaciones lingüísticas.

Desventajas:

- Computacionalmente intensivo.
- Requiere modelos pre entrenados como BERT.

6.16.2 BLEU (Bilingual Evaluation Understudy)

BLEU es una métrica tradicionalmente utilizada en la evaluación de traducción automática. Mide la precisión de n-gramas de la respuesta generada en comparación con la respuesta de referencia. (Chiusano, F. (2022, January 15))

Funcionamiento:

1. Cálculo de N-gramas:
    - Descompone las respuestas generadas y de referencia en n-gramas (secuencias de n palabras).
    - Generalmente, se consideran n-gramas de 1 a 4 palabras.
2. Precisión de N-gramas:
    - Calcula la precisión de n-gramas contando cuántos n-gramas de la respuesta generada aparecen en la respuesta de referencia.
    - Precisión de N-gramas:

$$Precisión = \frac{Número\ de\ n-gramas\ coincidentes}{Número\ total\ de\ n-gramas\ en\ la\ respuesta\ generada}$$

3. Penalización por Longitud:
    - Aplica una penalización por longitud para evitar que las respuestas cortas obtengan puntuaciones demasiado altas.
    - El cálculo del BP se calcula:

$$BP = \left\{ \begin{matrix} 1\ si\ c>r \\ e^{(1-\frac{r}{c})}\ si\ c\leq r \end{matrix} \right\}$$

Donde “c” es la longitud de la respuesta generada y “r” la longitud de la respuesta de referencia.

4. Cálculo de la Puntuación BLEU:
    - La puntuación BLEU final es el producto de la precisión de n-gramas y la penalización por longitud:

$$BLEU = BP \bullet exp(\sum_{n=1} w_n logP_n)$$

donde P_n es la precisión de los n-gramas y w_n son los pesos (generalmente iguales para todos los n-gramas).

Ventajas:

- Fácil de calcular e interpretar.
- Ampliamente utilizada y aceptada.

Desventajas:

- No captura bien la semántica y el contexto.
- Sensible a variaciones lingüísticas y sinónimos.

6.16.3 ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

ROUGE es una métrica utilizada principalmente para la evaluación de resúmenes automáticos y generación de texto. Mide la superposición de n-gramas, palabras y frases entre la respuesta generada y la referencia. (Chiusano, F. (2023, August 4))

Funcionamiento:

- ROUGE-1

Mide la superposición de unigrama (palabra individual) entre el resumen generado y el resumen de referencia. Es la forma más básica de ROUGE y se calcula como:

- Precision: Proporción de unigrama coincidentes en el resumen generado respecto al total de unigrama en el resumen generado.
- Recall: Proporción de unigrama coincidentes en el resumen generado respecto al total de unigrama en el resumen de referencia.
- F1 Score: Media armónica de la precisión y el recall.

- ROUGE-2

  Mide la superposición de bigramas (pares de palabras contiguas) entre el resumen generado y el resumen de referencia. Se calcula de manera similar a ROUGE-1 pero utilizando bigramas en lugar de unigramas:
  - Precision: Proporción de bigramas coincidentes en el resumen generado respecto al total de bigramas en el resumen generado.
  - Recall: Proporción de bigramas coincidentes en el resumen generado respecto al total de bigramas en el resumen de referencia.
  - F1 Score: Media armónica de la precisión y el recall.

- ROUGEL

  Mide la calidad del resumen considerando la Longitud de la Subsecuencia Común Más Larga (LCS, por sus siglas en inglés) entre el resumen generado y el resumen de referencia. La LCS toma en cuenta tanto el contenido como el orden de las palabras, capturando de manera más efectiva la estructura de las frases.

La LCS es la subsecuencia más larga que aparece en el mismo orden en ambas secuencias, pero no necesariamente de forma contigua. Por ejemplo, si se tiene:

- Texto A: "El gato negro duerme en el sofá".
- Texto B: "El gato está durmiendo en el sofá".

La LCS es "El gato en el sofá".

ROUGE-L utiliza la LCS para medir la precisión y el recall de la siguiente manera:

- Precisión (Precision): Proporción de la longitud de la LCS respecto a la longitud del resumen generado.

$$Precisión = \frac{Longitud\ de\ la\ LCS}{Longitud\ de\ la\ respuesta\ generada}$$

Ventajas:

- Evalúa la capacidad de capturar el contenido relevante y su estructura.
- Considera tanto precisión como recall.

Desventajas:

- Al igual que BLEU, puede no capturar completamente el contexto y el significado semántico.
- Puede ser sensible a la variación en la redacción.

Si por ejemplo se tiene un resumen de referencia y uno generado:

- Resumen de referencia: "El gato negro duerme en el sofá."
- Resumen generado: "El gato negro está durmiendo en el sofá."

- 6.16.5 ROUGE-Lsum

ROUGE-Lsum es una variante de ROUGE-L diseñada para evaluar resúmenes de múltiples oraciones o documentos enteros. En lugar de calcular la LCS para cada oración individualmente, ROUGE-Lsum considera la LCS a nivel de todo el documento o el resumen.

Diferencias con ROUGE-L

- ROUGE-L: Se enfoca en la LCS entre oraciones individuales del resumen generado y de referencia.
- ROUGE-Lsum: Calcula la LCS considerando todo el documento como una secuencia, lo cual es más adecuado para resúmenes largos o documentos completos.

El proceso es similar a ROUGE-L, pero la LCS se calcula sobre la concatenación de todas las oraciones en el documento o resumen.

Ejemplo:

- Resumen de referencia:
    - Oración 1: "El gato negro duerme en el sofá."
    - Oración 2: "Está muy cómodo."
- Resumen generado:
    - Oración 1: "El gato negro está durmiendo en el sofá."
    - Oración 2: "Se ve muy cómodo."

Se concatenan las oraciones y se calcula la LCS para las secuencias completas:

- Referencia concatenada: "El gato negro duerme en el sofá. Está muy cómodo."
- Generado concatenado: "El gato negro está durmiendo en el sofá. Se ve muy cómodo."

LCS = "El gato negro en el sofá muy cómodo."

- Longitud de la LCS = 8 palabras.
- Longitud del resumen generado = 12 palabras.
- Longitud del resumen de referencia = 11 palabras.

Para ROUGE-1:

- Unigrama coincidentes: ["El", "gato", "negro", "en", "el", "sofá"]
- Precision: 6/7 (unigrama en el generado)
- Recall: 6/7 (unigrama en el referencia)

Para ROUGE-2:

- Bigramas coincidentes: ["El gato", "gato negro", "en el", "el sofá"]
- Precision: 4/6 (bigramas en el generado)
- Recall: 4/6 (bigramas en el referencia)

Para ROUGE-L:

- Longitud de la LCS: 6 ("El gato negro en el sofá")
- Precision: 6/7
- Recall: 6/7

Para ROUGE-Lsum:

- Se calcularía la LCS para todo el documento en vez de oraciones individuales.

6.16.6 Wiki Split

Wiki Split recopila métricas como Sari, Exact y Sacrebleu, para evaluar modelos generadores de texto.

SARI (System output Against References and against the Input sentence):
SARI es una métrica diseñada específicamente para evaluar la calidad de la simplificación de texto. Evalúa la cantidad de palabras añadidas, eliminadas y mantenidas en la simplificación en comparación tanto con el texto original como con una referencia simplificada.

SARI mide la calidad de la simplificación en tres aspectos:

1. Additions (Adiciones): Evalúa las palabras nuevas que se agregan en la simplificación.
2. Deletions (Eliminaciones): Evalúa las palabras que se eliminan del texto original.
3. Keeps (Mantenimientos): Evalúa las palabras que se mantienen del texto original.

Para cada una de estas categorías, SARI calcula el precision, recall y F1 Score, y luego combina estos valores para obtener una puntuación global.

Ejemplo:

- Texto Original: "El rápido zorro marrón salta sobre el perro perezoso."
- Simplificación Generada: "El zorro marrón salta sobre el perro."
- Simplificación Referencia: "El zorro salta sobre el perro."
- Adiciones: Palabras en la simplificación generada pero no en el texto original.
- Eliminaciones: Palabras en el texto original pero no en la simplificación generada.
- Mantenimientos: Palabras que están tanto en el texto original como en la simplificación generada.

Exact Match:
Exact Match es una métrica sencilla que mide la proporción de oraciones generadas que coinciden exactamente con las oraciones de referencia.

Primero compara cada oración generada con la oración de referencia correspondiente. Si la oración generada es idéntica a la de referencia, cuenta como una coincidencia exacta.

Por ejemplo:

- Texto Original: "El rápido zorro marrón salta sobre el perro perezoso."
- Simplificación Generada: "El zorro marrón salta sobre el perro."
- Simplificación Referencia: "El zorro marrón salta sobre el perro."

En este caso, hay una coincidencia exacta, por lo que la puntuación de Exact Match sería alta.

SacreBLEU:
SacreBLEU es una versión estandarizada de la métrica BLEU, diseñada para evaluar la calidad de la traducción automática y la generación de texto. SacreBLEU se centra en reproducibilidad y consistencia en la evaluación.

SacreBLEU utiliza n-gramas para medir la precisión entre el texto generado y el texto de referencia. Incluye una penalización por longitud para evitar sesgos hacia textos más cortos. Se enfoca en la precisión de n-gramas (unigramas, bigramas, trigramas, etc.).

7. Metodología

En el proyecto, se ha optado por emplear un enfoque de investigación mixto, fundamentado en varias consideraciones. Entre ellas:

- La percepción del usuario con respecto a las respuestas generadas. El entendimiento del texto está intrínsecamente vinculado con la mente humana. Más allá de la precisión de las respuestas en relación con la información proporcionada, es esencial evaluar su claridad, coherencia y naturalidad.
- Se hace necesario utilizar métodos cuantitativos para evaluar la eficiencia técnica de las respuestas generadas y su precisión a gran escala.

El proyecto se ha dividido en las siguientes etapas:

- Investigación: En esta etapa, se recopilará información sobre aspectos clave, como los modelos que se utilizarán para las pruebas, las métricas cualitativas y cuantitativas más adecuadas, la necesidad de adquirir conocimientos adicionales para implementar la interfaz, y la mejor manera de interpretar los resultados obtenidos.

- Implementación: En esta fase, se desarrollará la interfaz que cargará los modelos seleccionados y permitirá mantener una conversación con un usuario. Además, se crearán las métricas cualitativas y cuantitativas para evaluar cada modelo.

- Pruebas: En este punto, la interfaz estará terminada y se procederá a probar los modelos con distintos usuarios, recopilando la información necesaria.

- Conclusiones: Finalmente, se analizará y se dará sentido a toda la información recopilada.

8. Desarrollo

En esta sección se explicará el proceso para crear la interfaz con la que realizaremos las pruebas de los modelos. Es importante recordar que esta es nuestra herramienta más importante, pero es solo eso, una herramienta.

Como se mencionó anteriormente, esta interfaz será capaz de leer archivos PDF y extraer su información. Con base en esa información, y utilizando los modelos de texto, el usuario podrá mantener una conversación. Las respuestas generadas serán evaluadas de manera cuantitativa mediante métricas computacionales específicas, las cuales también estarán integradas en el sistema.

Para efectos prácticos, en esta sección se referirá a toda la interfaz y sus funciones en general como el **"Sistema"**. Se denominará **"Usuario"** a la persona indistinta y sin conocimientos profundos sobre el proyecto que vaya a probarlo. Se le llamará **“Modelo”** a cualquier modelo de lenguaje en general que se mencione.

8.1 Modelos utilizados

Los modelos que se eligieron son modelos probados y respaldados por compañías grandes que pueden dar un patrón a seguir teniendo en cuenta lo importantes que son en el mundo de la IA actualmente. (Srivastava, T. (2024, January 8))

8.1.1 OpenAI GPT 3.5 Turbo

GPT-3.5 Turbo es una evolución dentro de la familia de modelos de lenguaje de OpenAI. La serie GPT (Generative Pre-trained Transformer) comenzó con GPT-1, seguido por GPT-2, GPT-3, y luego GPT-3.5, que incluye versiones como GPT-3.5 Turbo. Estos modelos se han desarrollado para mejorar progresivamente en términos de capacidad de comprensión y generación de lenguaje natural.

- GPT-1 (junio 2018): Demostró que un modelo pre entrenado en un gran corpus de texto y luego ajustado para tareas específicas podía superar los modelos anteriores en diversas tareas de procesamiento de lenguaje natural (NLP).
- GPT-2 (febrero 2019): Incrementó significativamente el tamaño del modelo, mostrando habilidades avanzadas en generación de texto y otras tareas de NLP.
- GPT-3 (junio 2020): Introdujo una escala aún mayor, con 175 mil millones de parámetros, lo que permitió una mejora notable en la generación de texto coherente y relevante.
- GPT-3.5 (2022-2023): Se desarrolló con mejoras técnicas y optimizaciones basadas en las lecciones aprendidas de GPT-3. Incluye variantes como GPT-3.5 Turbo.

GPT-3.5 Turbo está construido sobre la arquitectura de los transformadores, que es una tecnología de aprendizaje profundo. Los transformadores utilizan mecanismos de autoatención para procesar la entrada secuencialmente y generar salidas con una comprensión contextual avanzada.

Sus componentes más importantes son:

1. Capa de Entrada: Embeddings de palabras que convierten el texto en vectores numéricos.
2. Capas de Atención: Las capas de autoatención permiten que el modelo se enfoque en diferentes partes de la entrada para entender el contexto.

3. Capas de Feedforward: Redes neuronales totalmente conectadas que procesan la salida de las capas de atención.
4. Capas de Salida: Generan la secuencia de texto final.

Aunque OpenAI no ha revelado todas las especificaciones técnicas detalladas de GPT-3.5 Turbo, se pueden inferir algunas características basadas en la evolución de los modelos GPT anteriores y la información disponible públicamente.

- GPT-3.5 Turbo probablemente tiene una cantidad de parámetros similar o superior a GPT-3, que tiene 175 mil millones de parámetros.
- Entrenado en una vasta cantidad de texto que incluye libros, artículos, sitios web y otros recursos para capturar una amplia gama de información y estilos de escritura.
- Mejora en la eficiencia computacional y optimizaciones en la arquitectura del modelo para mejorar la velocidad de generación y la capacidad de manejar tareas complejas.

Algunas mejoras respecto a sus antecesores son:

- Rendimiento: GPT-3.5 Turbo es conocido por ser más rápido y eficiente en comparación con GPT-3. Ha sido optimizado para una menor latencia en la generación de texto.
- Precisión: Refinamiento en la capacidad del modelo para generar respuestas precisas y coherentes con menos "alucinaciones" o errores de información.

Comparado con otros modelos de lenguaje, GPT-3.5 Turbo destaca por su balance entre capacidad de generación de texto, rapidez y costo. Otros modelos de la competencia, como los desarrollados por Google, Microsoft y otros, también utilizan arquitecturas basadas en transformadores, pero la optimización y los ajustes específicos de OpenAI han hecho que GPT-3.5 Turbo sea una opción popular en el mercado.

A pesar de sus avances, GPT-3.5 Turbo tiene algunas limitaciones:

- El modelo no tiene acceso a eventos o información posterior a su fecha de corte de entrenamiento.
- Aunque avanzado, aún puede tener dificultades con contextos extremadamente complejos o ambiguos.
- Puede reflejar sesgos presentes en los datos de entrenamiento, por lo que es importante usarlo con precaución en aplicaciones sensibles.

8.1.2 Google Flan T5 Base

Flan-T5 es una versión mejorada del modelo de lenguaje T5 (Text-To-Text Transfer Transformer) desarrollado por Google. T5 ha sido uno de los modelos más influyentes en el procesamiento de lenguaje natural (NLP) debido a su enfoque en convertir todas las tareas de NLP en un formato de entrada/salida de texto a texto. Flan-T5 incorpora mejoras específicas de ajuste fino y tareas instructivas para mejorar aún más su rendimiento.

El modelo T5 fue introducido por Google Research en 2019 en el artículo "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". T5 se destacó por su enfoque unificado de tratamiento de tareas de NLP mediante el marco de texto a texto, lo que simplifica la arquitectura y facilita el entrenamiento y ajuste fino en una amplia variedad de tareas.

Flan-T5 se presentó posteriormente, incorporando métodos avanzados de ajuste fino mediante el uso de tareas instructivas (instruction tuning) y aprendizaje multitarea, con el objetivo de mejorar el rendimiento en tareas específicas y generales de NLP.

Flan-T5 se basa en la arquitectura T5 (Text-to-Text Transfer Transformer), la cual a su vez se basa en la arquitectura de transformers.

1. Utiliza capas de codificador y decodificador para procesar y generar texto.
2. Emplea autoatención para manejar la relación entre diferentes palabras en una secuencia y atención cruzada entre el codificador y el decodificador.
3. Entrenado en una amplia gama de tareas de NLP en un formato unificado de texto a texto.

La versión base de Flan-T5 contiene aproximadamente 220 millones de parámetros, lo que la sitúa en un punto medio en términos de tamaño y capacidad en comparación con otras versiones de T5.
Está entrenado en un corpus extenso y diverso de textos que incluye una amplia variedad de fuentes, como libros, artículos, sitios web y más.Además utiliza técnicas avanzadas de ajuste fino en tareas específicas para mejorar el rendimiento general y específico del modelo.

Una de las principales mejoras de Flan-T5 es el uso de tareas instructivas, donde el modelo se entrena con instrucciones específicas que mejoran su capacidad para seguir indicaciones y realizar tareas de manera más precisa.

El ajuste fino en múltiples tareas ayuda a Flan-T5 a generalizar mejor en tareas nuevas y no vistas durante el entrenamiento.

Además está optimizado para ser más eficiente en términos de tiempo de inferencia y uso de recursos computacionales.

Flan-T5 se destaca en comparación con otros modelos de lenguaje grandes debido a su enfoque en el ajuste fino mediante tareas instructivas, lo que mejora su rendimiento en una amplia gama de tareas de NLP. Comparado con modelos como GPT-3.5 Turbo, Flan-T5 ofrece un enfoque más estructurado para el entrenamiento y ajuste fino en múltiples tareas.

Aunque Flan-T5 ofrece numerosas ventajas, también presenta algunas limitaciones:

- Entrenar y desplegar modelos grandes como Flan-T5 base requiere recursos computacionales significativos.

- El rendimiento del modelo está limitado por la calidad y diversidad de los datos de entrenamiento. Sesgos en los datos pueden reflejarse en las respuestas generadas.

- Aunque es avanzado, aún puede tener dificultades con contextos extremadamente complejos o ambiguos.

8.1.2 MistralAI Mistral-7B-Instruct-v0.2

Mistral-7B-Instruct-v0.2 es un modelo de lenguaje desarrollado por la organización Mistral AI. Es un modelo de tamaño mediano con 7 mil millones de parámetros diseñado para tareas de instrucción y generación de texto natural.

Mistral-7B-Instruct-v0.2 se basa en la arquitectura de transformadores, similar a otros modelos de lenguaje como GPT y T5.

1. Encoders y Decoders: Emplea capas de codificador y decodificador para procesar entradas y generar salidas.
2. Autoatención: Utiliza mecanismos de autoatención para entender el contexto dentro de una secuencia de texto.
3. Atención Cruzada: Facilita la relación entre diferentes partes de la entrada y la salida.

Cuenta con 7 mil millones de parámetros, lo que lo coloca en la categoría de modelos medianos. Esto equilibra la capacidad de generación y la eficiencia computacional.

Está entrenado en un extenso corpus de datos que incluye una diversidad de textos provenientes de libros, artículos, sitios web, y otras fuentes.

Está ajustado específicamente para seguir instrucciones en lenguaje natural, mejorando su capacidad para manejar tareas dirigidas.

Sus características más fuertes son:

- Optimización de Instrucciones: Ajustado para seguir mejor las instrucciones dadas en lenguaje natural, lo que lo hace adecuado para aplicaciones que requieren respuestas precisas y contextuales.

- Eficiencia Computacional: Diseñado para ser más eficiente en términos de tiempo de inferencia y uso de recursos computacionales, permitiendo una implementación más práctica en diversas aplicaciones.

- Versatilidad: Capaz de manejar una variedad de tareas de NLP, desde la generación de texto hasta la respuesta a preguntas y la traducción.

Comparado con otros modelos de lenguaje como GPT-3.5 Turbo y Flan-T5, Mistral-7B-Instruct-v0.2 ofrece un enfoque especializado en tareas de instrucción, lo que lo hace particularmente útil para aplicaciones que requieren un seguimiento preciso de las indicaciones dadas en lenguaje natural. Su tamaño más reducido (7 mil millones de parámetros) lo hace más eficiente en términos de recursos computacionales, aunque podría tener limitaciones en comparación con modelos mucho más grandes en términos de capacidad de generación de texto y comprensión de contexto.

8.2 Backend

Ahora que se han seleccionado los modelos, se puede iniciar con el backend del Sistema. Para este fin, se ha decidido emplear Python como lenguaje de programación, ya que cuenta con una gran cantidad de librerías relacionadas con la IA, así como la implementación de distintos modelos.

Se utilizará FastAPI como framework, porque, aunque es relativamente nuevo, es poderoso, sencillo, ligero y fácil de utilizar. Esto es especialmente adecuado, ya que no se necesitan funciones complejas más allá de las librerías de Python disponibles.

8.2.1 Creación de server en FastAPi

Siguiendo la documentación oficial de FastAPi, lo primero que hay que hacer es instalar el paquete mediante:

```
$ pip install fastapi
```

Y también uvicorn, que ayudará a levantar el servidor:

```
$ pip install uvicorn
```

Y para tener nuestro el básico se necesita un archivo que se llamará app.py, el cual contendrá lo siguiente:

```
from fastapi import FastAPI
from routes.pdf import pdfRouter
from fastapi.middleware.cors import CORSMiddleware
app = FastAPI()

origins = ["*"]

app.add_middleware(
   CORSMiddleware,
   allow_origins=origins,
   allow_credentials=True,
   allow_methods=["*"],
   allow_headers=["*"],
)

app.include_router(pdfRouter , prefix="/pdf", tags=["pdf"])
```

Lo primero que se hace es importar FastAPI, un archivo llamado pdfRouter que es donde irán todas las rutas del servidor, y algo llamado CORSMiddleware, que son simplemente librerías por parte de FastAPI para hacer funcionar los cors.

Más abajo, se ve cómo es que se inicializa app, que es justamente la forma en que va a iniciarse el servidor, se le añaden los cors, y finalmente se le agregan las rutas, que se verán más adelante. Ya con eso se tiene el servidor básico de FastAPI, y para levantarlo simplemente se necesita correr en la terminal el comando:

```
$ uvicorn app:app –reload
```

8.2.2 HuggingFace

Los Modelos que se van a utilizar provienen, en su mayoría, de HuggingFace. ¿Qué es HuggingFace? Se podría decir que es un repositorio para proyectos de inteligencia artificial. En dicha plataforma se pueden encontrar documentación, implementaciones y el código de modelos de todo tipo, creados por la comunidad.

HuggingFace ofrece distintas formas de conectarse a un proyecto. Una de ellas es HuggingFaceHub, que se puede instalar mediante la terminal y se utiliza con una API Key, la cual es proporcionada una vez que se crea una cuenta en la plataforma. Cada modelo tendrá su propia documentación para su uso. En este caso, los Modelos que se utilizarán se importan de la siguiente manera:

```
HUGGINGFACE_API_KEY = os.getenv("HUGGINGFACE_API_KEY")
llm_openai = ChatOpenAI()
llm_flan = HuggingFaceHub(repo_id="google/flan-t5-base",
huggingfacehub_api_token=HUGGINGFACE_API_KEY)
llm_mistral = HuggingFaceHub(repo_id="mistralai/Mistral-7B-Instruct-v0.2",
huggingfacehub_api_token=HUGGINGFACE_API_KEY)
```

Como se puede ver, se está utilizando el Modelo Flat T5 Base de Google, Mistral 7B Instruct v0.2, y el modelo Gpt 3.5 Turbo de OpenAI, el cual no viene específicamente de HuggingFace, pero no presenta ningún inconveniente que no sea así.

```
lms = {
   "openai": llm_openai,
   "flan": llm_flan,
   "mistral": llm_mistral,
}
```

Luego, se van a poner los modelos en un objeto, para que se pueda acceder a cada uno de ellos de manera sencilla, simplemente poniendo algo como:

```
llms["openai"] #si se quiere acceder al modelo openai
```

También se puede crear un endpoint bastante útil que devuelva los Modelos que se tengan dentro, del objeto 'lms', que se usará para que desde el frontend se pueda elegir entre cada uno de ellos.

```
@pdfRouter.get("/llms")
async def get_all_llms():
   try:
       llms = get_llms()
       return {"llms": llms}
   except Exception as e:
```

```
       raise HTTPException(status_code=500, detail=str(e))
```

Y la función de 'get_llms' simplemente trae las llaves del objeto que contiene los modelos.

```
def get_llms():
   return list(llms.keys())
```

8.2.3 Subir un PDF

La primera cosa que se debe hacer es subir la información con la cual los Modelos van a trabajar. Esto será mediante archivos PDF, y el código que hará eso es el siguiente.

```
@pdfRouter.post("/upload")
async def upload_pdf(token: Annotated[str, Depends(oauth2_scheme)], file: UploadFile =
File(...)):
    try:
        text = extract_text_from_pdf(file.file)
        chunks = get_text_chunks(text)
        user = get_current_user(token)
        inserted_id = documents_collection.insert_one({"filename": file.filename,
"chunks": chunks, "user": ObjectId(user['id'])}).inserted_id
        return {"id": str(inserted_id), "text": text, "chunks": chunks}
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

Lo primero que se observa es que se trata de un endpoint de tipo POST con la ruta '/upload'. Este formato se repetirá en todas las rutas, por lo que se omitirá la explicación más adelante. La función solicita un token y un archivo de tipo UploadFile, proporcionado por FastAPI.

La variable 'text' contiene el texto extraído del PDF. Desglosando más detalladamente, la función extract_text_from_pdf es la siguiente:

```
def extract_text_from_pdf(file):
    reader = PdfReader(file)
    text = ""
    for page in reader.pages:
        text += page.extract_text()
    return text
```

El objeto 'PdfReader' proviene de un paquete llamado PyPDF y utiliza el archivo que se le proporcionó. Hay un ciclo que recorre página por página y va concatenando el texto de cada una, para finalmente devolver todo el texto junto.

Siguiendo con el código para subir el PDF, se encuentra una variable que se iguala a una función llamada 'get_text_chunks', la cual recibe el texto extraído del PDF. Esta función realiza lo siguiente:

```
def get_text_chunks(text: str):
    splitter = RecursiveCharacterTextSplitter(
```

```
        chunk_size=1000,
        chunk_overlap=200,
        length_function=len,
    )
    chunks = splitter.split_text(text)
    return chunks
```

Como se puede observar, primero se inicializa una variable con un objeto llamado 'RecursiveCharacterTextSplitter'. Este objeto devuelve una lista de chunks, que son partes más pequeñas de nuestro texto original, para que el modelo pueda procesarlo de mejor manera. (Eteimorde, Y. (2023, 16 agosto))

Volviendo a 'upload_pdf', la línea en la que se declara el 'user', lo que hace es buscar el usuario mediante el token proporcionado.

```
def get_current_user(token: Annotated[str, Depends()]):
    data = decode_access_token(token)
    user = userEntity(users_collection.find_one({"_id": ObjectId(data["sub"])}))
    return user
```

Cabe mencionar que se está utilizando MongoDB como base de datos. Aunque es una base de datos NoSQL, no es prioritario utilizar una base extremadamente específica, ya que principalmente se guardan datos numéricos y cadenas de texto.

Lo siguiente que se hace es insertar un registro en la colección llamada 'documents' de la base de datos. Los datos que se agregan son el nombre del archivo, la lista de chunks que se obtuvieron anteriormente y el ID del usuario.

Finalmente, se devuelve un objeto que contiene el ID del registro recién creado, el texto extraído del PDF y la lista de chunks.

8.2.4 Realizar una pregunta

La parte de realizar preguntas es quizá la más larga en cuanto a código, por eso es que se va a ir separando y revisando en partes más pequeñas.

```
@pdfRouter.post("/makeQuestion")
async def make_question(data: QuestionData):
    try:
        question = data.question
        doc = documentEntity(documents_collection.find_one({"_id": ObjectId(data.id)}))
        if doc is None:
```

```
            raise HTTPException(status_code=404, detail="Document not found")
        chunks = doc["chunks"]
        vector_store = get_vector_store(chunks)
        chat_history = conversations_collection.find_one({"document":
ObjectId(doc["id"])})
```

Se observa que se tiene un endpoint de tipo POST llamado '/makeQuestion', el cual solicita una variable llamada 'data' que es de tipo 'QuestionData'. Este tipo tiene la siguiente estructura:

```
class QuestionData(BaseModel):
    question: str
    id: str
    llm: str
    reference: str = None
```

Continuando con el código, lo primero que se hace es asignar la pregunta recibida a una variable llamada 'question'. Luego, se busca en la colección 'documents' el registro del documento al que se le va a hacer la pregunta, mediante su ID. Si no lo encuentra, devolverá una excepción. A continuación, se llama a una función llamada 'get_vector_store', la cual recibe la lista de chunks del documento y realiza lo siguiente:

```
def get_vector_store(chunks: list[str]):
    embeddings = OpenAIEmbeddings()
    vector_store = FAISS.from_texts(
        embedding=embeddings,
        texts=chunks,
    )
    return vector_store
```

En el marco teórico se discutieron los embeddings, que en pocas palabras, son el resultado de convertir el lenguaje humano en vectores matemáticos para que la computadora pueda procesarlos. En este caso, se usará el modelo de 'OpenAIEmbeddings' para obtener estos vectores, aunque se puede utilizar cualquier otra opción deseada.

Luego, se inicializará el 'vector_store'. FAISS proporciona la opción de guardar nuestros embeddings en su base de datos local, que está diseñada específicamente para alojar vectores. Ese 'vector_store' es lo que se va a devolver.

Más adelante lo que se hace es buscar el 'chat_history', que es donde se van a ir guardando los mensajes que se vayan mandando.

```
if chat_history is None:
```

```
            new_chat_history = save_conversation_chain(doc["filename"], doc["id"],
question)
            conversation_chain = get_conversation_chain_with_history(vector_store,
new_chat_history["chat_history"], data.llm)
            response = conversation_chain({"question": question})
            if data.llm == 'mistral':
                res = response["answer"].split('Helpful Answer:')[1]
            else:
                res = response["answer"]
            new_chat_history["chat_history"].append({"by": "ai", "text": res})
            conversations_collection.update_one({"file_name": doc["filename"]}, {"$set":
{"chat_history": new_chat_history["chat_history"]}})
            test_result = None
            if data.reference != '':
                test_result = await test_question(data.reference, vector_store,
new_chat_history, question)
            return {"chat_history": new_chat_history["chat_history"], "test":
test_result}
```

Si no encuentra 'chat_history', significa que es el primer mensaje que se manda en ese documento. Entonces, llamamos a 'save_conversation_chain', que va a guardar la conversación y va a devolver la conservación que se acaba de guardar.

```
def save_conversation_chain(file_name: str, document: str, question: str):
   inserted_id = conversations_collection.insert_one({"file_name": file_name,
"document": ObjectId(document), "chat_history": [
       {
           "by": "user",
           "text": question,
       }
   ]}).inserted_id
   chat_history = conversations_collection.find_one({"_id": inserted_id})
   return chat_history
```

Ahora, se va a llamar a la función 'get_conversation_chain_with_history', la cual pide el 'vector_store', el 'chat_history', el 'llm' (que se refiere al modelo que estamos utilizando y que especificaremos desde el frontend) y una variable booleana para determinar si estamos realizando un test, la cual también será enviada desde el frontend.

```
def get_conversation_chain_with_history(vector_store, chat_history: list,
llm:str):
   try:
       memory = ConversationBufferMemory(
           memory_key="chat_history",
           vector_store=vector_store,
           similarity_threshold=0.8,
           max_memory_size=1000000,
           return_messages=True,
           input_key="question",
```

```
        )

        for chat in chat_history:
            if chat["by"] == "user":
                memory.chat_memory.add_user_message(chat["text"])
            else:
                memory.chat_memory.add_ai_message(chat["text"])
        conversation_chain = ConversationalRetrievalChain.from_llm(
            retriever=vector_store.as_retriever(),
            llm=llms[llm],
            memory=memory
        )
        return conversation_chain
    except Exception as e:
        raise e
```

Primero se debe inicializar una variable memory con un objeto ‘ConversationBufferMemory’. Este objeto, proveniente de LangChain, ayudará a mantener una conversación. Si no se utiliza esto, cada petición que se hiciera hacia el modelo sería completamente indiferente al historial de mensajes enviados y recibidos. Este objeto pide algunos parámetros, como ‘memory_key’, que debe ser "chat_history" para guardar los mensajes que se vayan agregando. En ‘vector_store’ se pondrá el vector store que se ha enviado a la función. El ‘similarity_threshold’ se refiere a qué tan libremente puede generar respuestas aleatorias el modelo y es un valor de 0 a 1. En ‘max_memory_size’, como su nombre lo indica, se va a establecer el tamaño máximo de memoria que va a aceptar la conversación, así que se pondrá un valor bastante grande porque no se sabe cuántos mensajes tendrá cada conversación. El ‘input_key’ es simplemente para indicar que le estaremos haciendo preguntas, por lo que su valor será "question".

Luego se iterará en el ‘chat_history’, que es una lista de mensajes. Se irán agregando al memory los mensajes generados por el usuario o por el modelo, según corresponda.

Con ‘memory’ completamente inicializada, se creará un objeto de tipo ‘ConversationRetrivalChain’, que también proviene de LangChain. Esto permitirá que el modelo responda preguntas teniendo cierto contexto, por eso le pasamos como retriever el ‘vector_store’ (que es el arreglo numérico que representa el texto), así como la memory y el llm que vamos a utilizar para generar las respuestas.

Con esto, ya se puede empezar a generar respuestas, como se ve en la variable response, que devolverá un objeto con el ‘chat_history’, la respuesta, nombre del modelo y otros datos.

Ahora se procederá a colocar la respuesta generada por el modelo en la variable res. La condición para elegir el modelo Mistral es específica porque este genera un objeto distinto a los demás.

Luego, se agrega el nuevo mensaje generado por el modelo al 'chat_history', y se actualizará el registro de la conversación actual con el nuevo 'chat_history'.

Después, se tiene una condición que verifica si la referencia recibida es diferente a una cadena vacía. Si está vacía, significa que no se está en modo de test, lo cual se evaluará desde el frontend. Dentro de esa condición, se llamará a una función que realizará un test pidiendo la referencia, el 'vector_store', el historial del chat y la pregunta original. Esto probará las respuestas generadas por cada modelo para dar el resultado de las métricas, pero eso se detalla más adelante.

Continuando con el código de '/makeQuestion', esta es la parte restante:

```
else:
            chat_history["chat_history"].append({"by": "user", "text": question})
            conversations_collection.update_one({"file_name": doc["filename"]},
{"$set": {"chat_history": chat_history["chat_history"]}})
            conversation_chain =
get_conversation_chain_with_history(vector_store, chat_history["chat_history"],
data.llm, False)
            response = conversation_chain({"question": question})
            if data.llm == 'mistral':
                res = response["answer"].split('Helpful Answer:')[1]
            else:
                res = response["answer"]
            chat_history["chat_history"].append({"by": "ai", "text": res})

            conversations_collection.update_one({"file_name": doc["filename"]},
{"$set": {"chat_history": chat_history["chat_history"]}})
            test_result = None
            if data.reference != '':
                test_result = await test_question(data.reference, vector_store,
chat_history, question)
            return {"chat_history": chat_history["chat_history"], "test":
test_result}
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

Este código es bastante similar al que se ejecutaba cuando no existía 'chat_history'. La única diferencia es que aquí no se crea un nuevo chat, sino que se utiliza el que ya existe.

Al final, en ambos casos, se devuelve el 'chat_history' y el resultado del test. Si no está en modo de pruebas, el test simplemente se devolverá como valor nulo.

8.2.5 Testear el modelo con métricas computacionales

Anteriormente se observó que, en ciertos casos, se llama a una función llamada 'test_question' al momento de realizar una pregunta. A continuación, se detalla lo que hace esta función:

```
@pdfRouter.post("/testQuestion")
async def test_question(reference: str, vector_store, chat_history, question: str):
   results = []
   for llm in llms:
       conv = get_conversation_chain_with_history(vector_store,
chat_history["chat_history"], llm)
       res = conv({"question": question})
       if llm == 'mistral':
           prediction = res["answer"].split('Helpful Answer:')[1]
       else:
           prediction = res["answer"]
       prediction_list = [prediction]
       reference_list = [reference]
       bleu_precision = get_bleu_score(predictions=prediction_list,
reference=reference_list)
       bert_score = get_bertscore_score(predictions=prediction_list,
reference=reference_list)
       rouge_score = get_rouge_score(predictions=prediction_list,
reference=reference_list)
       results.append({"llm": llm, "bleu": bleu_precision, "bert": bert_score, "rouge":
rouge_score})
   return results
```

Esta función requiere una cadena de texto llamada 'reference', que es un string proporcionado por el usuario con la respuesta que cree que el Modelo va a generar (considerando que estamos en modo de pruebas, por lo que se supone que el usuario debería tener una idea o noción de una respuesta acorde a la información del PDF proporcionado).

Se declara una lista de resultados, donde se irán agregando los resultados de cada modelo. Luego, se iterará en la lista de Modelos que se tengan, se obtendrá el conversation chain (con la misma función que se llama cuando se hace una pregunta) y, en base a eso, se le pasará el input del Usuario. Lo que devuelva será la respuesta del Modelo, y se evaluará esa respuesta con las métricas que fueron seleccionadas.

Las métricas seleccionadas son formas matemáticas de medir la precisión en general de los modelos. Claramente, existe la posibilidad de hacer estas mediciones manualmente y utilizando muchas matemáticas, pero en HuggingFace hay algunas formas de hacerlo de manera automatizada mediante el software que proporciona la comunidad.

Para poder utilizar esto, primero se necesita instalar un paquete de HuggingFace llamado evaluate.

```
$ pip install evaluate
```

De ese paquete, se tiene que importar una función llamada ‘load’.

```
from evaluate import load
```

Y para cargar las métricas, se debe poner el nombre de cada una, el cual viene en su misma documentación.

```
#https://huggingface.co/spaces/evaluate-metric/bleu
bleu = load('bleu')

#https://huggingface.co/spaces/evaluate-metric/bertscore
bertscore = load('bertscore')

# https://huggingface.co/spaces/evaluate-metric/rouge
```

```
rouge = load('rouge')
```

Se puede crear una función para cada una de ellas para conseguir los resultados que se obtienen , de la siguiente forma:

```
def get_bleu_score(predictions: list[str], reference:list[str]):
   results = bleu.compute(predictions=predictions, references=reference)
   return results

def get_perplexity_score(predictions: list[str]):
   results = perplexity.compute(predictions=predictions, )
   return results

def get_bertscore_score(predictions: list[str], reference:list[str]):
   results = bertscore.compute(predictions=predictions, references=reference, lang="en")
   return results

def get_rouge_score(predictions: list[str], reference:list[str]):
   results = rouge.compute(predictions=predictions, references=reference)
   return results
```

Y dichas funciones son las que se utilizan cada vez que se quiera hacer una prueba de un modelo. Recapitulando, lo que se hizo fue recorrer cada uno de los modelos cargados, hacerles la pregunta formulada por el usuario y realizar pruebas con las métricas seleccionadas, agregando los resultados a una lista. Esta lista será devuelta como resultado de la función.

Si se desea ver el backend del proyecto más a detalle y terminado, se puede ingresar a la siguiente liga: https://github.com/ERICKGALVAN/pdf_chat_backend

8.3 Frontend

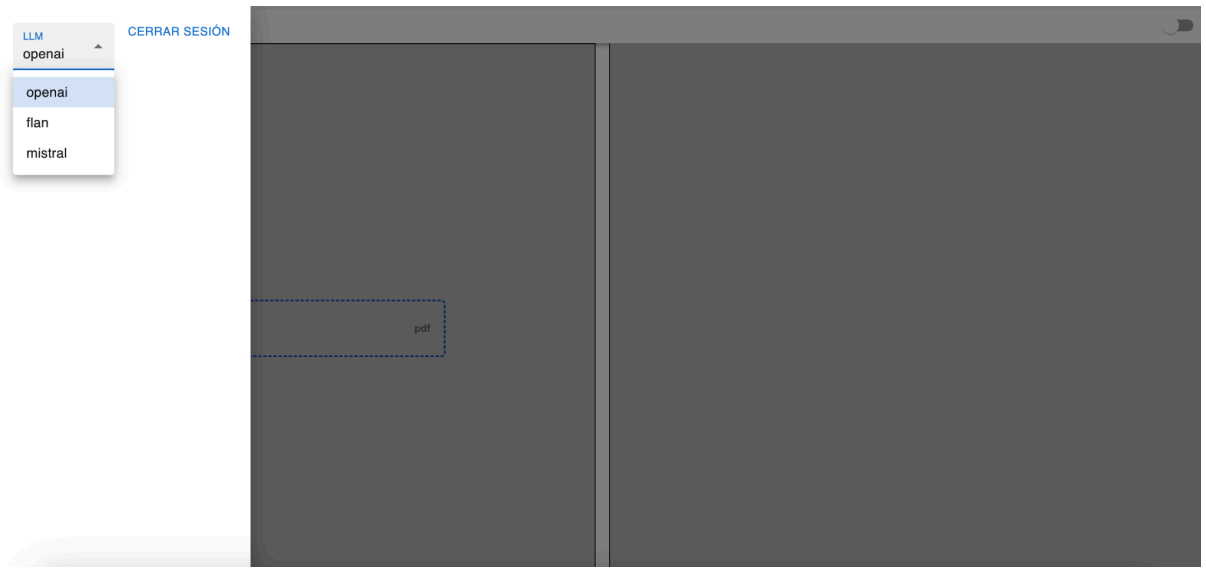
Una vez que se han implementado todas las funciones para subir un PDF, procesar su información, hacer preguntas referentes a ella y testearlas, se puede proceder a crear la interfaz de usuario.

Se ha decidido utilizar React con TypeScript para tener un mayor control en el manejo de tipos de datos. En esta sección se mostrará la lógica para manejar todos los datos en general, y poco o nada de componentes y elementos que conforman la UI del sistema. Por lo tanto, también se omite mostrar la configuración básica de la aplicación y entrar en profundidad en aspectos como el manejo de estado, rutas, etcétera.

8.3.1 Subir el archivo PDF

“Captura de pantalla 1 del Sistema “

Se crea una interfaz bastante sencilla como la anterior. En la esquina superior izquierda, hay un botón para abrir un menú. En la esquina superior derecha, hay un interruptor que se usará para cambiar entre el modo de pruebas y el modo normal. La mitad izquierda de la pantalla contiene un botón que servirá para subir el archivo PDF. En la mitad derecha aparecerá el chat una vez que se haya iniciado una conversación.



“Captura de pantalla 2 del Sistema “

En el menú desplegable (drawer) se encontrarán el botón de cerrar sesión (se está trabajando con usuarios y tokens, pero se omite el proceso de inicio de sesión y creación de usuarios ya que no tiene mayor relevancia) y un botón de selección, que mostrará los posibles modelos a utilizar. Estos modelos se obtienen desde el endpoint configurado en el backend, y el que se elija será el que generará las respuestas que se mostrarán en el chat, pero cuando se está en modo de pruebas no importa realmente cuál esté seleccionado, ya que las pruebas se hacen con todos los modelos, como se programó en la función para hacer tests en el backend. Por default tiene asignado el primero que aparezca en la lista, en este caso es el de OpenAI.

Cuando se hace clic en el botón para seleccionar un archivo PDF, simplemente se guarda el PDF localmente con la siguiente función.

```
const changeFile = (file?: File | null) => {
  setIsNew(true);
  setFile(file ?? null);
};
```

La variable ‘isNew’ se usa para determinar si el archivo es nuevo y evitar errores, como la carga duplicada del archivo. Una vez verificado esto, se puede llamar a la siguiente función cuando el archivo se haya cargado correctamente, para subirlo al backend.

```
function onDocumentLoadSuccess({ numPages }: { numPages: number }) {
  setNumPages(numPages);
  if (pdfContext.isNew) pdfContext.upLoadPdf(pdfContext.file!);
}
```

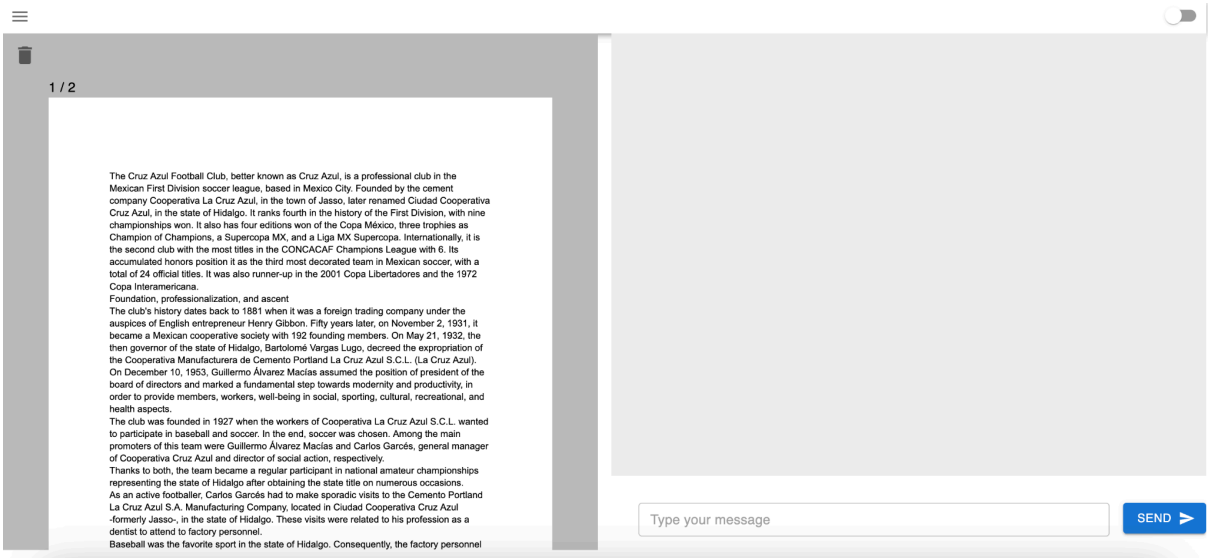
La variable ‘numPages’ ‘se utiliza para mostrar el archivo en el frontend, mientras que la función ‘upLoadPdf’ llama al endpoint /upload para subir el archivo.

```
async function upLoadPdf(pdf: File) {
 try {
   const formData = new FormData();
   formData.append("file", pdf);
   const response = await api.post("/pdf/upload", formData, {
     headers: {
       "Content-Type": "multipart/form-data",
     },
   });
   return response.data;
 } catch (err) {
   throw err;
 }
}
```

Y se inicializan algunas variables para comenzar a trabajar con el chat.

```
 const uploadPdf = async (file: File) => {
   setIsLoading(true);
   const doc = await pdfService.upLoadPdf(file);
   setCurrentDocument(doc);
   setChat([]);
   setIsLoading(false);
 };
```

Quedando de la siguiente manera una vez se ha subido un archivo correctamente:



“Captura de pantalla 3 del Sistema “

8.3.2 Hacer preguntas

Ahora que el archivo se ha cargado correctamente, está disponible para interactuar con él. Como se explicó anteriormente al crear el endpoint para hacer preguntas, este admite un dato opcional llamado ‘reference’. Si se incluye este dato, el sistema estará en modo de prueba; si no se incluye, estará en modo normal.

Para controlar esto, se utiliza el interruptor (switch) ubicado en la esquina superior derecha. Si el interruptor no está activo, el sistema estará en modo normal, y la variable ‘reference’ será nula o vacía al enviarse al backend. Si el interruptor está activo, se abrirá un segundo campo de entrada (input) donde el usuario podrá escribir el valor de la variable ‘reference’. Ambos campos de entrada deben estar validados para evitar el envío de valores vacíos, previniendo así cualquier tipo de error o comportamiento no deseado.

La función ‘makeQuestion’ queda de la siguiente forma:

```
const makeQuestion = async (question: string, reference: string | null) => {
   setChat((prevChat) => [...prevChat, { by: "user", text: question }]);
   setIsThinking(true);
   const data = await pdfService.makeQuestion(
     question,
     currentDocument.id,
     currentLlm!,
     reference ?? null
   );
   const aiMessage = data["chat_history"][data["chat_history"].length - 1];
   setChat((prevChat) => [...prevChat, { by: "ai", text: aiMessage["text"] }]);
   setIsThinking(false);
 };
```

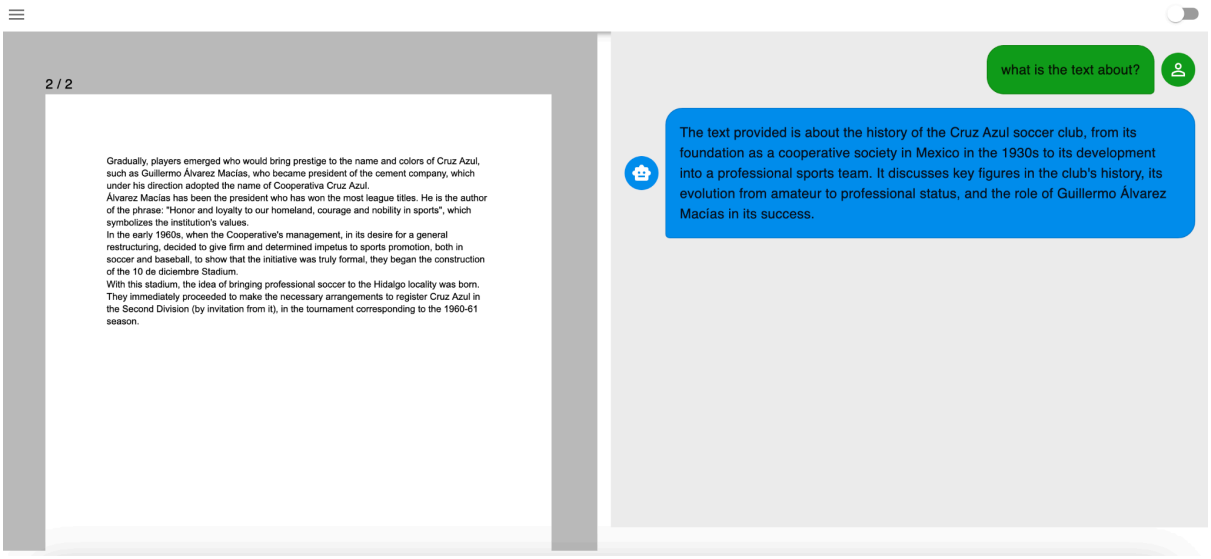
Como se puede observar, primero se solicita la pregunta ('question') y la referencia ('reference'), que puede ser nula. Luego, se actualiza la lista de mensajes agregando el mensaje recién añadido por el Usuario. Posteriormente, se llama al endpoint '/makeQuestion', al cual se envían la pregunta, el ID del documento PDF al que se hace referencia, el modelo actual (que es solo para el chat pero no relevante en modo de prueba) y la referencia. Luego, se agrega el mensaje devuelto por el modelo a la lista de mensajes que se mostrarán en pantalla.

Con esto, se puede visualizar el historial de mensajes y las respuestas del Modelo seleccionado. Es importante destacar que las pruebas y todo el proceso se realizarán en inglés para evitar posibles diferencias en las capacidades de traducción de cada modelo. Esto se hace teniendo en cuenta que el objetivo es evaluar solamente las respuestas a las preguntas planteadas.

Con eso, la respuesta recibida es la siguiente:

```
{
    'chat_history': [
        {
            'by': 'user',
            'text': 'what is the text about?'
        },
        {
            'by': 'ai',
            'text': "The text provided is about the history of the Cruz Azul
soccer club, from its foundation as a cooperative society in Mexico in the 1930s
to its development into a professional sports team. It discusses key figures in
the club's history, its evolution from amateur to professional status, and the
role of Guillermo Álvarez Macías in its success."
        }
    ],
    'test': None
}
```

Como se puede notar, Lo que devuelve es una lista de objetos en ‘chat_history’, en la que cada elemento es un mensaje, que lleva el remitente (‘user’ para el Usuario, y ‘ai’ para el Modelo), además de una propiedad ‘test’, que por el momento su valor es ‘None’ o nulo, ya que no enviamos el mensaje en modo de pruebas.



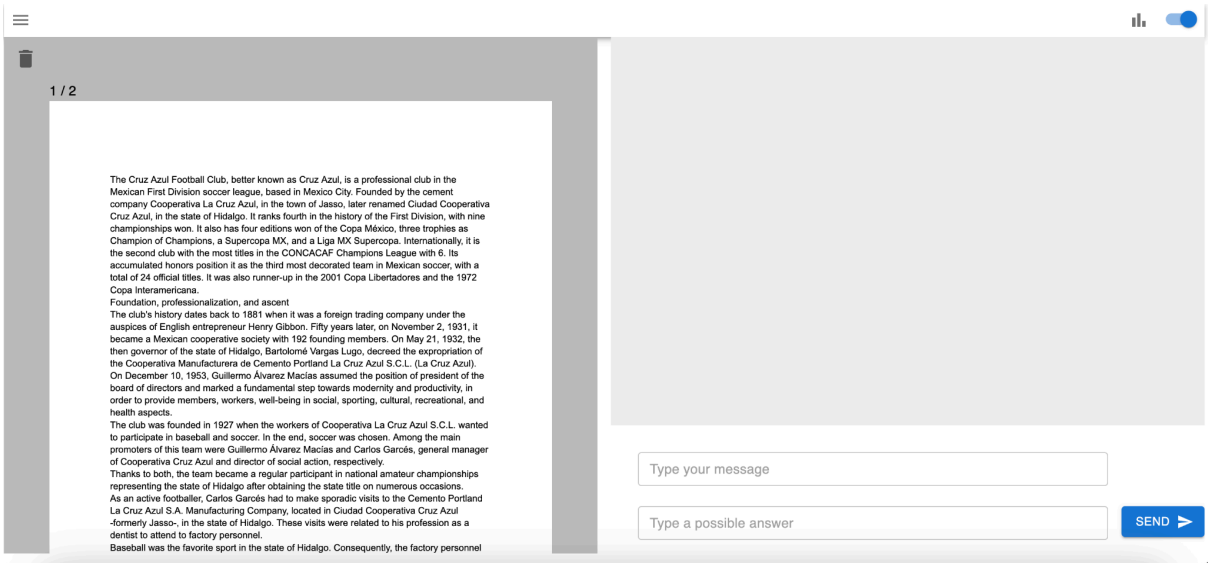
“Captura de pantalla 4 del Sistema “

8.3.3 Testear las respuestas

Con el chat en funcionamiento, lo siguiente es mostrar los resultados de las pruebas de las respuestas generadas por los modelos. A pesar de que en la interfaz de usuario solo se muestran las respuestas de un modelo, desde el backend se realizarán pruebas en todos.

Por lo tanto, cuando el interruptor esté activado, se abrirá un segundo campo de entrada donde el usuario debe escribir una referencia. La "referencia" es lo que el Usuario cree que debería ser una respuesta similar a lo que el modelo va a proporcionar. Por ejemplo, considerando el PDF proporcionado en las imágenes anteriores, si se pregunta "¿En qué año fue fundado Cruz Azul?", se esperaría una respuesta similar a "Cruz Azul fue fundado en 1927". Por eso, es crucial que el usuario que realiza las pruebas tenga un conocimiento del contenido del texto en el archivo PDF. En general, esta referencia es lo que las métricas utilizarán para determinar qué tan acertada es la predicción del Modelo según los estándares humanos.

Así es como se ve con los dos inputs activos cuando está el modo de pruebas:



“Captura de pantalla 5 del Sistema “

Donde el primer input se refiere a la pregunta o mensaje hacia el Modelo, y el segundo es la referencia. También se nota que cuando se activa al modo prueba aparece otro botón en la esquina superior derecha, el cual se usará para abrir un modal en el cual se irán escribiendo los resultados de las pruebas.

Ahora, para hacer las pruebas se había mencionado que se utilizaría la misma función con la que se hacen las preguntas, solamente que esta vez se enviará una cadena de texto como ‘reference’, en lugar de un valor nulo.

Haciendo eso, la respuesta que se obtiene es algo como esto:

```
{
    'chat_history': [
        {
            'by': 'user',
```

```
            'text': 'what is the text about?'
        },
        {
             'by': 'ai',
             'text': 'The text discusses the history and foundation of the Cruz
Azul football club, detailing its origin as a cooperative society and its
development into a professional soccer team. It also mentions key figures
involved in its formation and the values associated with the institution.'
        }
    ],
    'test': [
    {
        'llm': 'openai',
        'bleu':
        {
            'bleu': 0.0,
            'precisions': [0.011363636363636364, 0.0, 0.0, 0.0],
            ‘brevity_penalty': 1.0,
            'length_ratio': 17.6,
            'translation_length': 88,
            'reference_length': 5
        },
        'bert': {
            'precision': [0.8048691749572754],
            'recall': [0.8441911339759827],
            'f1': [0.824061393737793],
            'hashcode':
            'roberta-large_L17_no-idf_version=0.3.12(hug_trans=4.39.2)'
        },
        'rouge': {
            'rouge1': 0.07058823529411765,
            'rouge2': 0.024096385542168672,
            'rougeL': 0.04705882352941177,
            'rougeLsum': 0.04705882352941177
        }
    },
    {
        'llm': 'flan',
        'bleu': {
            'bleu': 0.0,
            'precisions': [0.0, 0.0, 0.0, 0.0],
            'brevity_penalty': 0.01831563888873418,
            'length_ratio': 0.2,
            'translation_length': 1,
            'reference_length': 5
        },
        'bert': {
            'precision': [0.8763363361358643],
            'recall': [0.794913649559021],
            'f1': [0.833641529083252],
            'hashcode':
```

```
'roberta-large_L17_no-idf_version=0.3.12(hug_trans=4.39.2)'
        },
        'rouge': {
            'rouge1': 0.0,
            'rouge2': 0.0,
            'rougeL': 0.0,
            'rougeLsum': 0.0
        },
    },
    {
        'llm': 'mistral',
        'bleu': {
            'bleu': 0.0,
            'precisions': [
                0.043478260869565216,
                0.022222222222222223,
                0.
                0,
                0.
                0
            ],
        'brevity_penalty': 1.0,
        'length_ratio': 9.2,
        'translation_length': 46,
        'reference_length': 5
        },
        'bert': {
            'precision': [0.8326574563980103],
            'recall': [0.8681427240371704],
            'f1': [0.8500299453735352],
            'hashcode':
'roberta-large_L17_no-idf_version=0.3.12(hug_trans=4.39.2)'
        },
        'rouge': {
            'rouge1': 0.16666666666666666,
            'rouge2': 0.08695652173913042,
            'rougeL': 0.08333333333333333,
            'rougeLsum': 0.08333333333333333
        }
     }
   ]
}
```


Además del chat_history, esta respuesta ahora incluye la propiedad test, que contiene cada una de las evaluaciones de las métricas para cada modelo cargado.

De esta manera, se puede verificar si la propiedad test existe y, de ser así, almacenar la información para mostrarla al usuario. La forma más conveniente de hacerlo es mostrar un promedio de los valores obtenidos para cada pregunta realizada. Esto es importante porque algunas respuestas pueden contener errores tipográficos, de contexto, etcétera. Al proporcionar un resumen promedio, a medida que se hagan más preguntas en la conversación, el margen de error disminuirá, logrando así un resultado más preciso y representativo.

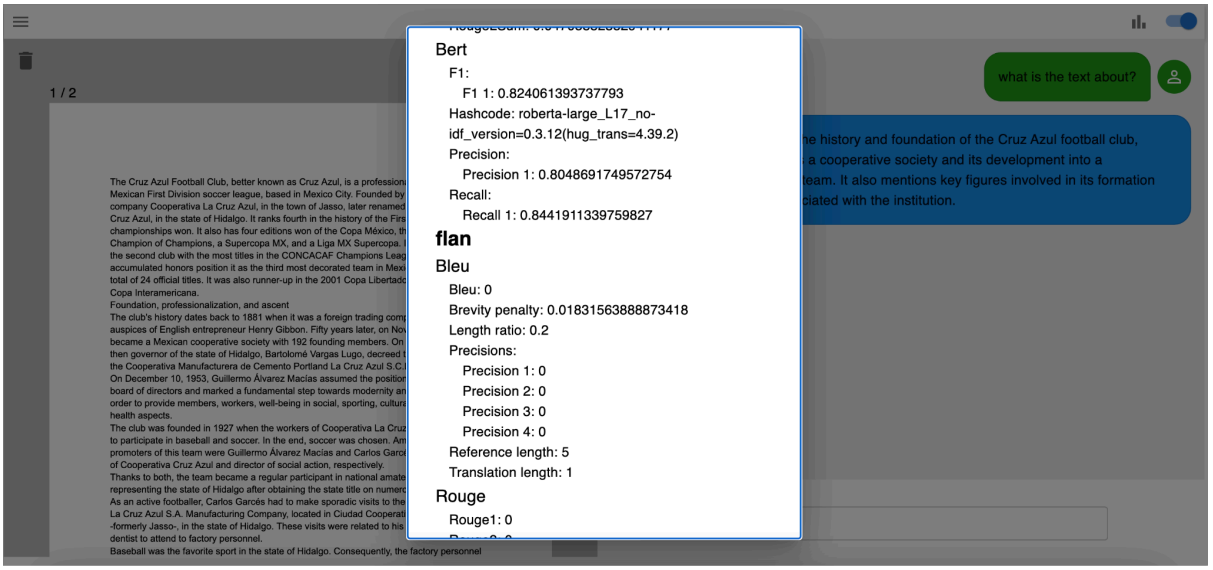
El siguiente código muestra la función makeQuestion que se presentó anteriormente, con la adición de una funcionalidad para guardar la información de las pruebas y calcular el promedio en cada pregunta.

```
const makeQuestion = async (question: string, reference: string | null) => {
   setChat((prevChat) => [...prevChat, { by: "user", text: question }]);
   setIsThinking(true);
   const data = await pdfService.makeQuestion(
     question,
     currentDocument.id,
     currentLlm!,
     reference ?? null
   );
   if (data["test"]) {
     const test = data["test"];
     if (testInfo) {
       setTestInfoAux(test);
       testInfo.forEach((element) => {
         if (element.llm === test.llm) {
           element.bleu.bleu = (element.bleu.bleu + test.bleu.bleu) / 2;
           element.bleu.brevity_penalty =
             (element.bleu.brevity_penalty + test.bleu.brevity_penalty) / 2;
           element.bleu.length_ratio =
             (element.bleu.length_ratio + test.bleu.length_ratio) / 2;
           element.bleu.reference_length =
             (element.bleu.reference_length + test.bleu.reference_length) / 2;
           element.bleu.translation_length =
             (element.bleu.translation_length + test.bleu.translation_length) /
             2;
           element.bleu.precisions = element.bleu.precisions.map(
             (value, index) => (value + test.bleu.precisions[index]) / 2
           );
           element.bert.f1 = element.bert.f1.map(
             (value, index) => (value + test.bert.f1[index]) / 2
           );
           element.bert.precision = element.bert.precision.map(
             (value, index) => (value + test.bert.precision[index]) / 2
           );
           element.bert.recall = element.bert.recall.map(
             (value, index) => (value + test.bert.recall[index]) / 2
```

```
          );
          element.bert.hashcode = test.bert.hashcode;
          element.rouge.rouge1 =
            (element.rouge.rouge1 + test.rouge.rouge1) / 2;
          element.rouge.rouge2 =
            (element.rouge.rouge2 + test.rouge.rouge2) / 2;
          element.rouge.rougeL =
            (element.rouge.rougeL + test.rouge.rougeL) / 2;
          element.rouge.rougeLsum =
            (element.rouge.rougeLsum + test.rouge.rougeLsum) / 2;
        }
      });
    }
    setTestInfo(test);
    setNumberTest(numberTest + 1);
  }

  const aiMessage = data["chat_history"][data["chat_history"].length - 1];
  setChat((prevChat) => [...prevChat, { by: "ai", text: aiMessage["text"] }]);
  setIsThinking(false);
};
```

Y visualmente se puede mostrar de la siguiente manera:



“Captura de pantalla 6 del Sistema “

En este link se puede encontrar el proyecto frontend terminado: https://github.com/ERICKGALVAN/pdf-chat

8.4 Métricas cualitativas

Es cierto que, desde un punto de vista técnico, tener datos numéricos sobre la precisión de los modelos es muy importante. Sin embargo, también es crucial contar con métricas cualitativas y conceptos comprensibles desde la perspectiva del usuario común. Uno de los principales objetivos de un modelo de lenguaje, además de proporcionar respuestas precisas, es imitar la forma de expresión humana de la mejor manera posible. Esto implica formular respuestas que hagan que el usuario se pregunte si está conversando con otra persona o con una máquina.

Es por eso que se han elegido ciertas métricas que ayudarán a calificar la experiencia humana al momento de utilizar los modelos, los cuales se mencionan a continuación.

8.4.1 Coherencia

Hablar de coherencia se refiere a la capacidad de un texto para presentar ideas de manera fluida y consistente de acuerdo al contexto. En este caso, el contexto está conformado por la información del archivo PDF subido y el historial de mensajes que se han enviado. (Farías, G. (2024, January 16))

La coherencia es fundamental para mantener una conversación de cualquier tipo, ya que pueden existir respuestas técnicamente correctas en cuanto a datos, pero que no sigan el contexto de la plática. Esto puede confundir al usuario y dificultar la comprensión del mensaje.

En este punto, podemos preguntarnos si la coherencia es igual para todos. Por ejemplo, si se hace la pregunta: “¿De qué color es la gorra de Alexis?”, un modelo podría contestar:

a) Negra
b) La gorra de Alexis es negra

Aunque ambas respuestas son correctas, ¿todas las personas estarían de acuerdo en cuál es más coherente? Es cierto que cada persona puede y debe tener su propia percepción de coherencia, lo cual es beneficioso al momento de hacer las pruebas. Cuantos más puntos de vista se consideren, más beneficioso será para el resultado final. Sin embargo, es importante especificar ciertos puntos a tener en cuenta al evaluar las respuestas.

1. La respuesta es lógica y no muestra contradicciones.
2. La respuesta es relevante y clara en el contexto de la pregunta.
3. La respuesta es clara y sin ambigüedades.

Con estos criterios, el usuario puede tener una idea más estándar de lo que se va a considerar como “coherente”.

Esta métrica se evaluará en una escala del 1 al 5, y la interpretación de los resultados se manejaría de la siguiente manera:

- Promedio de 4 a 5: Generalmente, las respuestas son coherentes, claras y contextualmente correctas.
- Promedio de 3: Las respuestas son aceptables, pero se pueden mejorar en cuanto a detalle y estructura.
- Promedio de 1 a 2: Hay problemas significativos en cuanto a la coherencia de la respuesta, como respuestas fuera de contexto o con lógica deficiente.

8.4.2 Relevancia

La relevancia se refiere a la capacidad de recibir una respuesta precisa y con información relevante acorde a la pregunta planteada, evitando información innecesaria. (SLU, Z. C. (n.d.))

Una respuesta relevante da la sensación de satisfacción al obtener justo la información buscada por el usuario, creando eficiencia (ahorrando tiempo y esfuerzo) y credibilidad, aumentando la confianza del usuario hacia el modelo.

Por ejemplo, ante la pregunta “¿Cuál es la capital de México?”, un modelo puede generar las siguientes respuestas:

a) La capital de México es Ciudad de México.
b) México es un país de América.

La respuesta a) proporciona la información buscada, por lo que se considera relevante. La respuesta b), aunque ofrece información sobre el país mencionado, no es relevante porque no responde a la pregunta.

Los puntos a considerar para evaluar si una respuesta es relevante son los siguientes:

1. La respuesta aborda directamente la pregunta o el tema planteado.
2. La respuesta da toda la información necesaria para responder a la pregunta.
3. La información proporcionada es exacta.

Las calificaciones irán en una escala de 1 a 5:

- Promedio de 4 a 5: Las respuestas generalmente son completas y precisas.

- Promedio de 3: Las respuestas generalmente son relevantes, pero pueden mejorarse en cuanto a exactitud o completitud.
- Promedio de 1 a 2: Las respuestas suelen ser irrelevantes, no abordan correctamente el tema planteado y contienen información inexacta.

8.4.3 Fluidez

La fluidez se refiere a la manera en que un texto es más sencillo de entender y procesar, gracias al correcto uso gramatical y su naturalidad. Esto es fundamental para mantener una buena legibilidad y una experiencia agradable para el usuario. Una falta de fluidez puede resultar cansada y hacer que se perciba más artificial al modelo. (Jimenez, A. (2019, February 28))

Por ejemplo:

a) A Fátima le gusta ver caricaturas.
b) Fátima le gusta ver caricaturas.

Ambos ejemplos son coherentes y gramaticalmente correctos, pero el ejemplo a) con una sola palabra puede cambiar radicalmente la forma en que se lee la oración, ya que su estructura es más común en una conversación cotidiana.

Para evaluar la fluidez, se pueden considerar los siguientes puntos:

1. Verificar qué tan fácil es leer y comprender la respuesta.
2. Asegurarse de que no haya errores gramaticales graves que alteren significativamente el significado y la comprensión de la oración.

Se utiliza una escala de 1 a 5 para medir los resultados:

- Promedio de 4 a 5: Las respuestas son fluidas y fácilmente comprensibles.
- Promedio de 3: Las respuestas son aceptables pero podrían mejorar en fluidez.
- Promedio de 1 a 2: Existen problemas significativos de fluidez y comprensión en las respuestas.

8.4.4 Creatividad

Aunque un modelo de este tipo no necesariamente debe estar diseñado para ser "creativo" y responder cosas que no estén directamente relacionadas con la pregunta planteada, esta métrica va a medir la originalidad o la capacidad de dar respuestas más allá de lo solicitado. Algunos modelos

pueden tener esta capacidad, mientras que otros no, por lo que este punto se considera un plus y nos permite examinar qué características permiten a un modelo ser más creativo. (Gil, J. M. (2018))

Por ejemplo, si se plantea la pregunta "¿Qué opinas del texto?"

a) Creo que es importante ser muy conscientes del impacto que estamos teniendo en el planeta y que el calentamiento global es un problema grave.
b) El calentamiento global es algo malo.
c) No sé.

d) El calentamiento global es el fenómeno en el cual las temperaturas globales se elevan como consecuencia de la contaminación generada por el ser humano.

Las cuatro opciones son completamente diferentes. Las respuestas a) y b) expresan opiniones reales sobre el tema, aunque la primera es más creativa que la segunda. Ambas pueden considerarse creativas en cierta medida.

Por otro lado, las respuestas c) y d) no muestran creatividad al abordar la pregunta. Mientras que la respuesta c) simplemente evade la pregunta, la respuesta d) parece no entenderla y ofrece información no solicitada.

Esta métrica nos permite entender mejor la capacidad de un modelo para emitir opiniones y determinar qué es necesario para que lo haga.

Para evaluar la creatividad de una respuesta, se deben considerar los siguientes puntos:

- Originalidad y enfoque único de la respuesta.
- Elementos inesperados o sorprendentes en la respuesta.
- Emoción o impresión que genera la respuesta en el usuario.

La evaluación se realiza en una escala del 1 al 5:

- Promedio de 4 a 5: Las respuestas muestran cierto grado de creatividad, ya sea simple o más elaborada.
- Promedio de 3: Las respuestas tienen una creatividad leve, pero podrían ser más innovadoras.
- Promedio de 1 a 2: Las respuestas carecen de creatividad, siguiendo patrones convencionales o proporcionando información no relevante para la pregunta planteada.

9 Experimentación y pruebas

Para evaluar los modelos, se realizarán pruebas con diez usuarios distintos. Habrá tres documentos PDF diferentes con información relacionada a la historia, el deporte y temas científicos. Dos de ellos serán utilizados por tres personas cada uno, y otro por 4 personas, de manera completamente aleatoria, llevando a cabo una conversación hasta completar 30 respuestas. Cada respuesta será calificada por el usuario según las métricas cualitativas y, al final, se apuntará el promedio de las métricas cuantitativas generadas.

El usuario deberá haber leído el documento antes de iniciar la prueba para facilitar el proceso de realizar preguntas y evaluar las respuestas.

En cuanto a las preguntas que debe realizar el usuario, tiene total libertad para escribir lo que desee. Sin embargo, se recomienda que empiece hablando sobre el tema del PDF asignado. La única condición es que lo que escriba tenga total coherencia y sentido para evitar respuestas que puedan resultar poco útiles por parte del modelo.

Todo el proceso se va a realizar en inglés, ya que de esta forma se obtiene un resultado más estándar y que evalúa únicamente las respuestas generadas y no su capacidad de traducción. Si alguno de los usuarios tiene problemas al entender el idioma, se le apoyará brindándole su traducción.

A continuación se muestran los resultados de todos los experimentos.

9.1 OpenAI Gpt 3.5 Turbo

9.1.1 Resultados cuantitativos

Participante	1	2	3	4	5	6	7	8	9	10	Promedio
Bleu											
Bleu	0.14 05	0.06 7	0	0.045	0.065	0.055	0.070	0.050	0.090	0.108	0.069 05
Brevity Penalty	1	1	1	1	1	1	1	1	1	1	1
Precision 1	0.25 07	0.15 9	0.16 5	0.180	0.170	0.210	0.200	0.190	0.195	0.190 3	0.191

Precision 2	0.1745	0.110	0.088	0.1692	0.090	0.150	0.110	0.130	0.100	0.120	0.13427
Precision 3	0.1287	0.071	0.039	0.050	0.1169667	0.060	0.095	0.095	0.085	0.080	0.091
Precision 4	0.091	0.043	0	0	0.030	0.035	0.032	0.036	0.033	0.035	0.0335
Reference length	7.648	7.437	9.125	7.50	8.74	7.75	8.50	7.80	8.20	8.00	8.032
Result length	40.1054	52.9668	49.75	45.0	48.7546	46.5	49.5	47.0	49.0	48.5	47.6078
Rouge											
Rouge1	0.383	0.275	0.288	0.300	0.35733	0.310	0.350	0.320	0.340	0.330	0.315333
Rouge2	0.256	0.205	0.205	0.220	0.239	0.225	0.245	0.230	0.240	0.235	0.2r21
RougeL	0.382	0.269	0.288	0.275	0.311	0.285	0.325	0.295	0.315	0.305	0.313
RougeLSum	0.383	0.269	0.288	0.275	0.3183	0.285	0.325	0.295	0.315	0.305	0.3133
Bert											
F1	0.892	0.854	0.848	0.860	0.8327	0.870	0.875	0.880	0.865	0.890	0.86467
Precision	0.861	0.832	0.829	0.840	0.8697	0.845	0.865	0.850	0.860	0.855	0.84067
Recall	0.925	0.876	0.867	0.870	0.82533	0.880	0.920	0.890	0.910	0.900	0.889333
WikiSplit											
Exact	0	0	0	0	0	0	0	0	0	0	0
Sacrebleu	15.044	8.551	5.842	6.000	11.68633	8.000	13.68633	10.000	14.000	12.000	9.812333

Sari	36.305	27.672	24.580	26.000	31.633	28.000	35.133	30.000	34.000	32.000	29.519

Resultados cuantitativos GPT

9.1.2 Resultados cualitativos

Participante	1	2	3	4	5	6	7	8	9	10	Promedio
Coherencia	4.3	4.57	4.6	4.83	4.23	4.64	4.42	4.95	4.56	4.8	4.69
Relevancia	4	4.57	5	4.8	5	4.34	4.87	4.7	4.3	4.65	4.623
Fluidez	4.6	4.57	4.8	5	4.7	4.9	4.64	4.7	4.78	4.8	4.886
Creatividad	3.4	4	3.6	4.3	4.7	3.5	3.8	3.2	4.1	3.7	3.83

Resultados cualitativos GPT

9.2 Google Flan T5 Base

9.2.1 Resultados cuantitativos

Participante	1	2	3	4	5	6	7	8	9	10	Promedio
Bleu											
Bleu	0.423	0	0.204	0.200	0.438	0.250	0.375	0.300	0.400	0.350	0.209
Brevity Penalty	0.884	0.094	0.901	0.200	0.68433	0.300	0.700	0.400	0.600	0.500	0.626333
Precision 1	0.604	0.015	0.630	0.300	0.6143	0.400	0.800	0.500	0.700	0.600	0.41633
Precision 2	0.455	0	0.5	0.200	0.62833	0.300	0.650	0.400	0.550	0.450	0.318333
Precision 3	0.39	0	0.15	0.200	0.288	0.300	0.250	0.400	0.450	0.350	0.184

	9		3								
Precision 4	0.308	0	0.125	0.190	0.250	0.160	0.130	0.190	0.200	0.290	0.1443
Reference length	7.648	7.437	9.125	6.000	5.490	7.000	11.000	8.000	10.000	9.000	8.07
Result length	8.292	1.625	9.75	5.000	0.88967	6.000	10.000	7.000	9.000	7.000	6.555667
Rouge											
Rouge1	0.655	0.020	0.569	0.300	0.60267	0.400	0.800	0.500	0.700	0.600	0.414667
Rouge2	0.473	0.015	0.416	0.275	0.294	0.350	0.180	0.425	0.335	0.250	0.3013
RougeL	0.655	0.020	0.569	0.400	0.803	0.500	0.900	0.600	0.800	0.700	0.414667
RougeLSum	0.655	0.020	0.569	0.476	0.567	0.593	0.281	0.279	0.317	0.389	0.414667
Bert											
F1	0.924	0.823	0.892	0.769	0.892	0.634	0.547	0.720	0.501	0.18766	0.879666
Precision	0.925	0.831	0.897	0.749	0.603	0.845	0.521	0.472	0.690	0.54833	0.884333
Recall	0.924	0.815	0.887	0.749	0.603	0.845	0.521	0.472	0.690	0.53733	0.41
WikiSplit											
Exact	0.390	0	0	0.390	0	0.010	0.140	0.190	0.240	0.080	0.13
Sacrebleu	42.337	0	21.086	10.000	25.000	35.000	20.000	15.000	17.987	25.000	21.141
Sari	34.731	21.657	26.268	45.252	25.13	40.657	25.986	15.718	17.429	25.43	28.7248

Resultados cuantitativos Flan

9.2.2 Resultados cualitativos

Participante	1	2	3	4	5	6	7	8	9	10	Promedio
Coherencia	2.4	1.29	2.8	3	2.6	2.5	2.6	1.9	1.45	1.8	2.274
Relevancia	2.5	1.71	3.3	2.6	1.9	3.8	3.2	3.6	2.7	2.9	2.851
Fluidez	1.8	1.29	2.6	1.5	2.1	1.7	1.9	2.2	1.8	2.9	1.986
Creatividad	1.4	1.14	1.8	1.3	1.9	1.2	1.5	1.9	1.3	1.9	1.528

Resultados cualitativos Flan

9.3 MistralAI Mistral-7B-Instruct-v0.2

9.3.1 Resultados cuantitativos

Participante	1	2	3	4	5	6	7	8	9	10	Promedio
Bleu											
Bleu	0.092	0.008	0	0.100	0.080	0.050	0.040	0.030	0.010	0.020	0.043
Brevity Penalty	1	1	1	1	1	1	1	1	1	1	1
Precision 1	0.139	0.099	0.094	0.300	0.025	0.200	0.025	0.100	0.078	0.050	0.113
Precision 2	0.109	0.053	0.009	0.250	0.0005	0.100	0.0005	0.030	0.004	0.015	0.0571
Precision 3	0.087	0.020	0.005	0.100	0.0025	0.080	0.0055	0.040	0.010	0.020	0.03705
Precision 4	0.071	0.005	0	0.100	0.050	0.020	0.010	0.0035	0.0025	0.001	0.0263

Reference length	7.648	7.437	9.125	15.000	1.489	12.000	4.000	10.000	6.000	8.000	8.0699
Result length	48.738	61.87	48.375	90.000	0.957	80.000	20.000	70.000	50.000	60.000	52.994
Rouge											
Rouge1	0.230	0.198	0.147	0.600	0.0023	0.300	0.020	0.200	0.040	0.200	0.18033
Rouge2	0.157	0.102	0.059	0.900	0.062	0.800	0.200	0.600	0.300	0.400	0.358
RougeL	0.230	0.167	0.144	0.400	0.052	0.080	0.050	0.200	0.060	0.100	0.126
RougeLSum	0.230	0.167	0.144	0.600	0.0023	0.300	0.020	0.200	0.040	0.100	0.18033
Bert											
F1	0.878	0.847	0.835	0.83	0.72	0.68	0.21	0.53	0.83	0.98	0.7371
Precision	0.846	0.826	0.821	0.83	0.94	0.84	0.87	0.82	0.97	0.88	0.8530
Recall	0.913	0.869	0.850	0.93	0.83	0.83	0.876	0.82	0.87	0.86	0.8807
WikiSplit											
Exact	0	0	0	0	0	0	0	0	0	0	0
Sacrebleu	9.834	3.541	1.602	5.48	8.567	4.87	5.27	3.87	6.88	6.34	5.553
Sari	31.722	21.536	20.898	32.54	20.42	28.44	31.4	24.54	28.5	26.5	28.555

Resultados cuantitativos Mistral

9.3.2 Resultados cualitativos

Participante	1	2	3	4	5	6	7	8	9	10	Promedio
Coherencia	4.8	4.86	4.8	4.9	4.7	4.6	4.9	4.5	5	4.9	4.806
Relevancia	4.6	4.82	5	4.9	4.4	4.6	4.9	4.6	4.3	4.3	4.712
Fluidez	4.7	4.84	4.4	4.9	4.6	4.9	4.5	4.6	5	4.5	4.734
Creatividad	4.2	4.57	4	3.9	4.9	4.5	4.3	4.9	4.3	4.6	4.427

Resultados cualitativos Mistral

9.4 Gráficas de los resultados

A continuación se muestran gráficas de los valores obtenidos en las métricas más importantes, siendo los valores en el eje X simplemente el número del participante por lo cual no importa su orden, mientras que el eje Y representa el valor obtenido.

9.4.1 GPT 3.5 Turbo

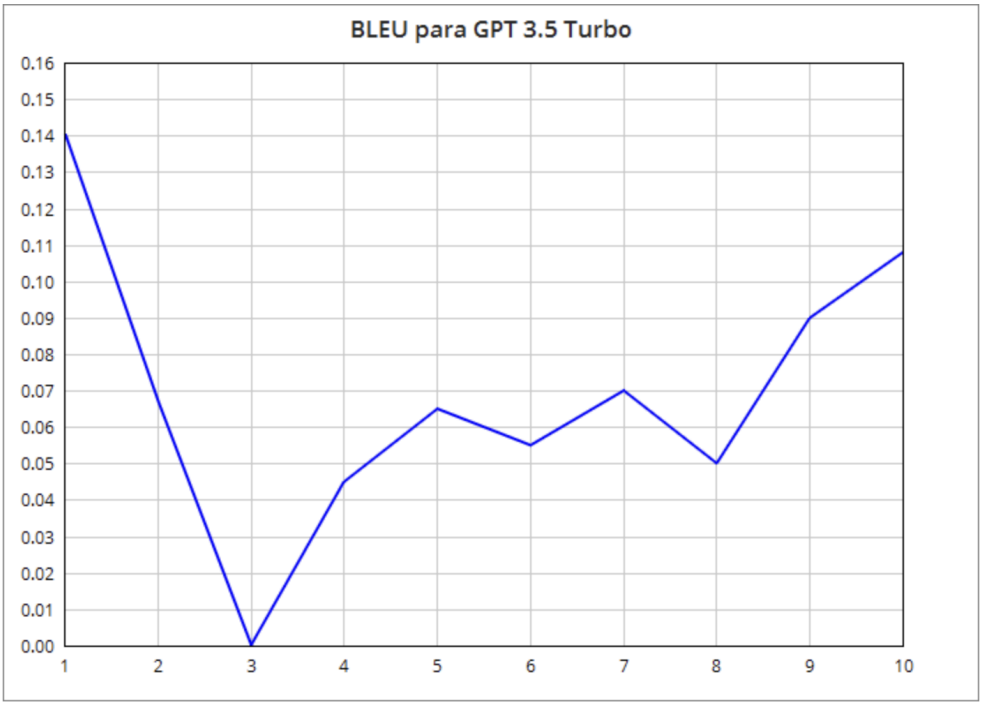


Gráfico BLEU para GPT

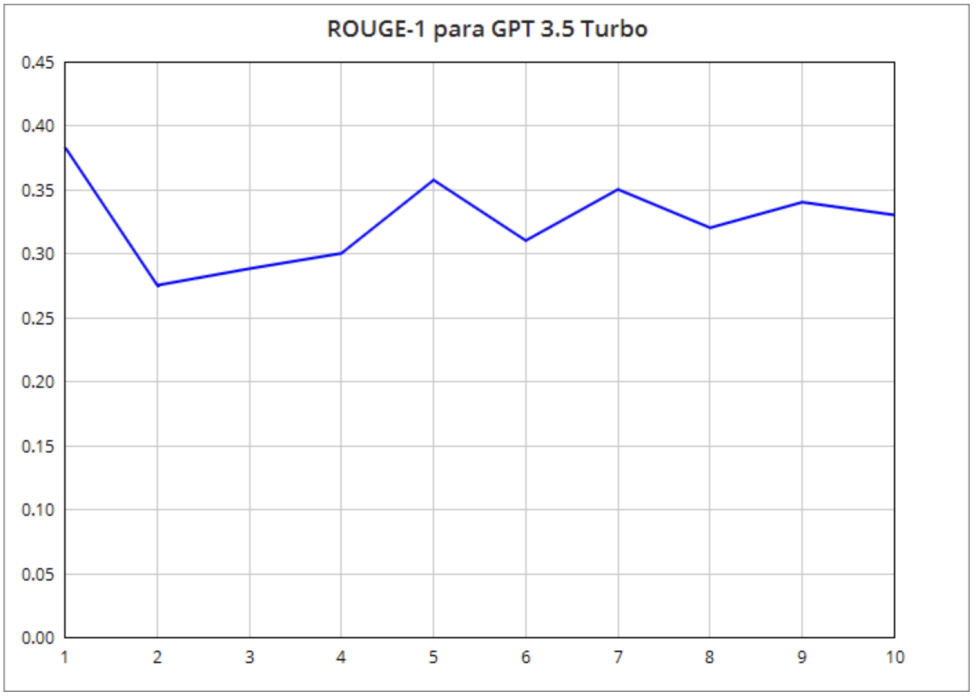


Gráfico ROUGE-1 para GPT

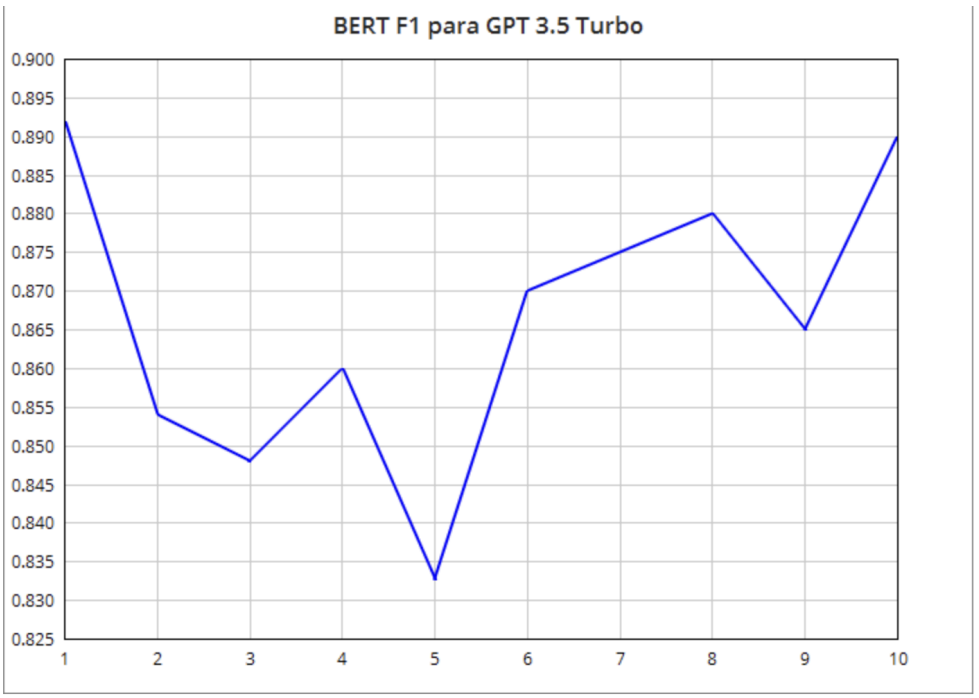


Gráfico BERT F1 para GPT

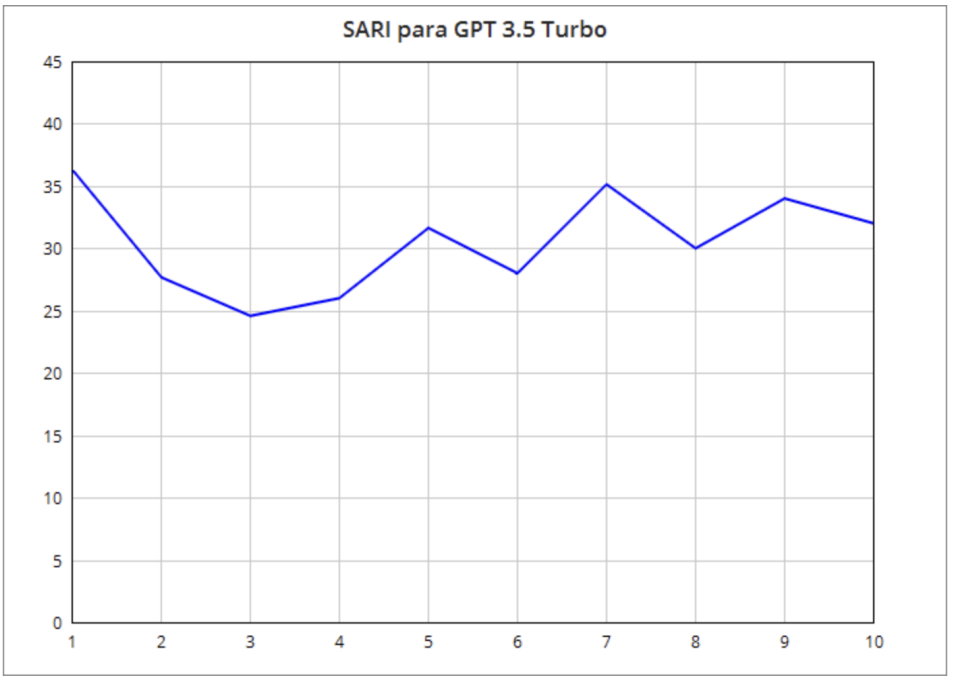


Gráfico SARI para GPT

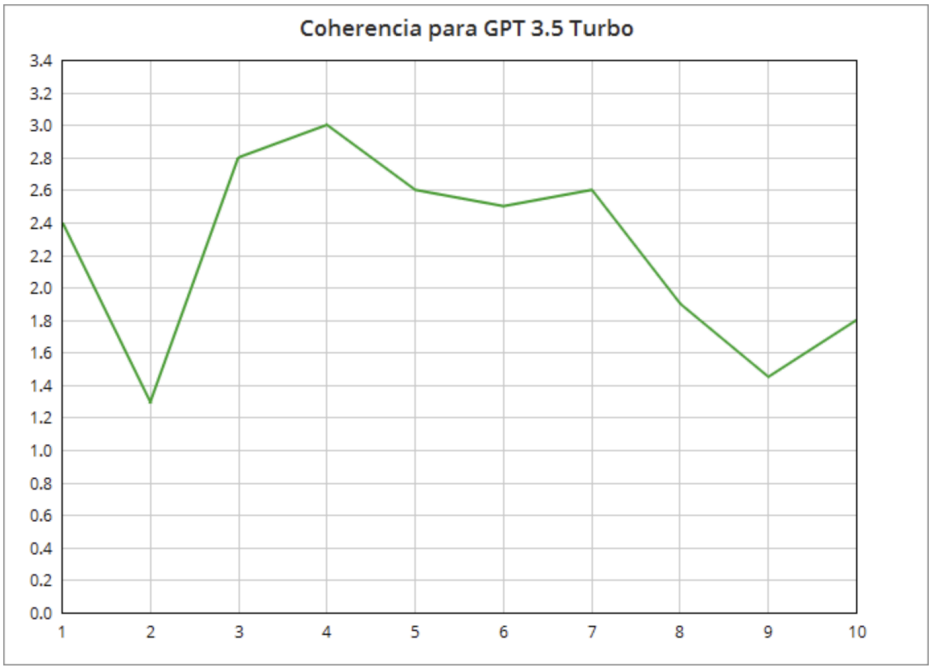


Gráfico coherencia para GPT

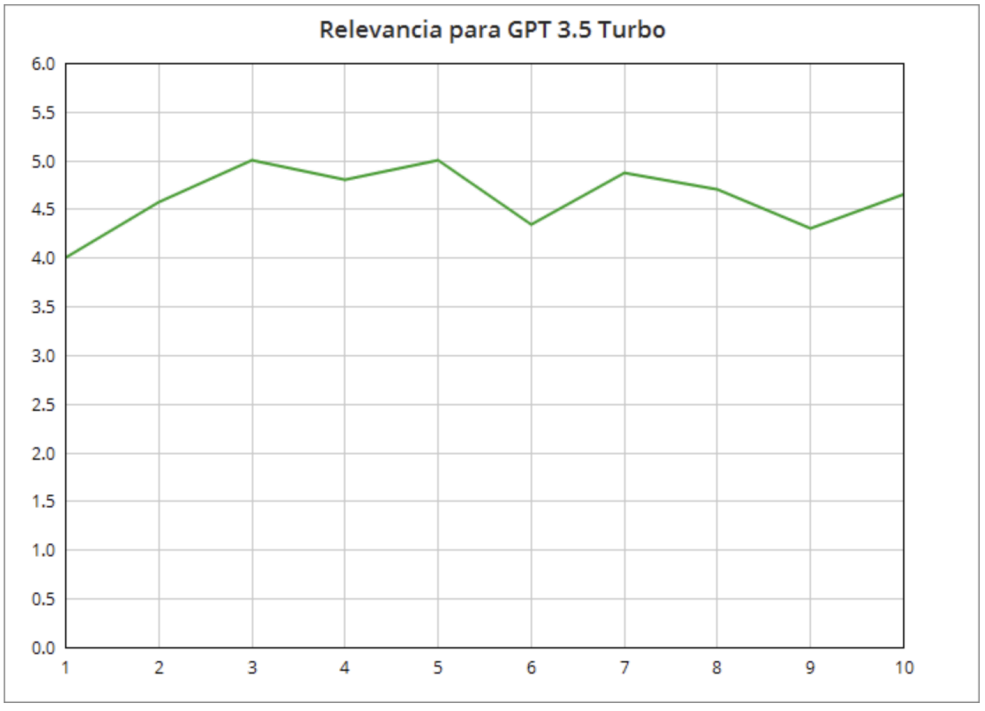


Gráfico Relevancia para GPT

9.4.2 Flan T5 Base

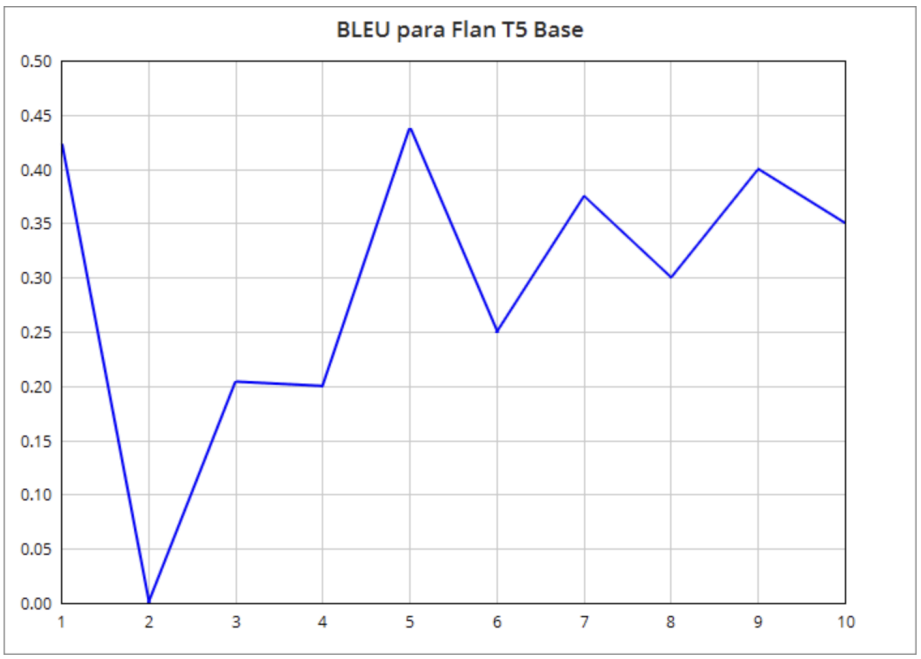


Gráfico BLEU para Flan

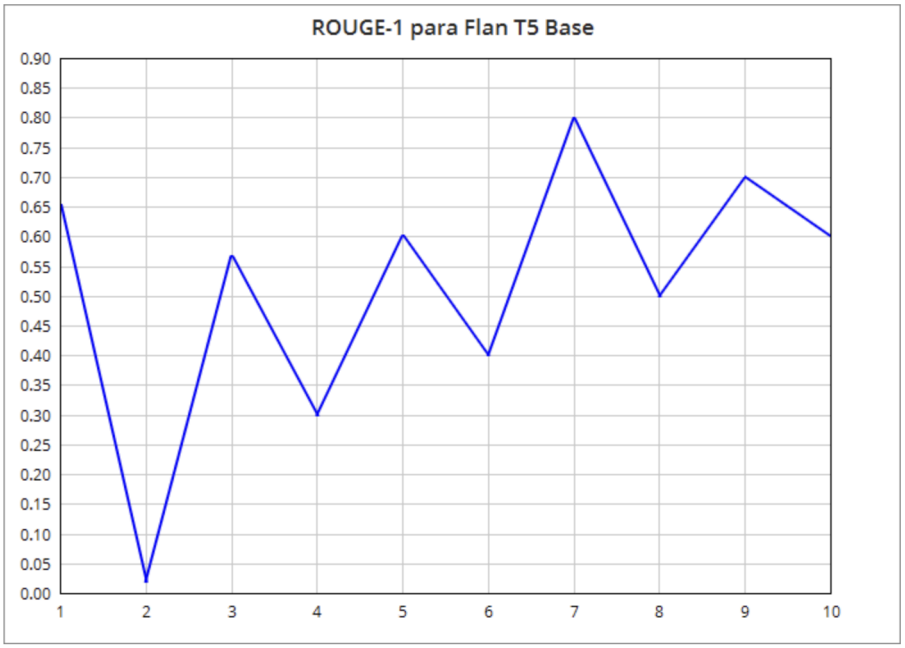


Gráfico ROUGE-1 para Flan

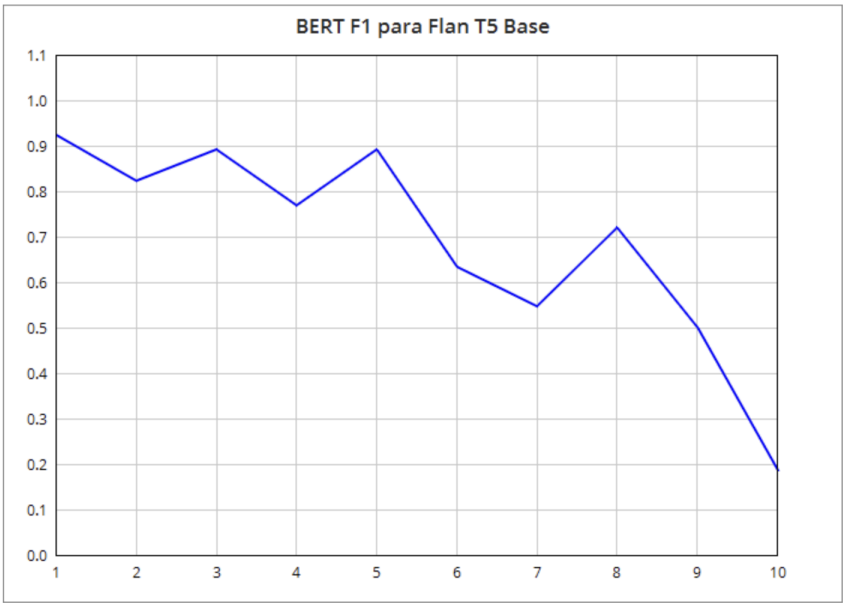


Gráfico BERT F1 para Flan

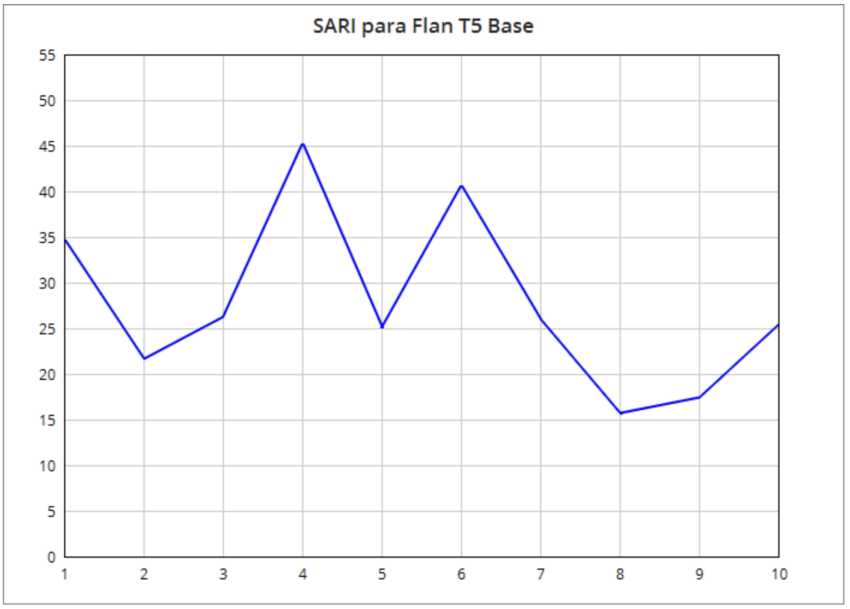


Gráfico SARI para Flan

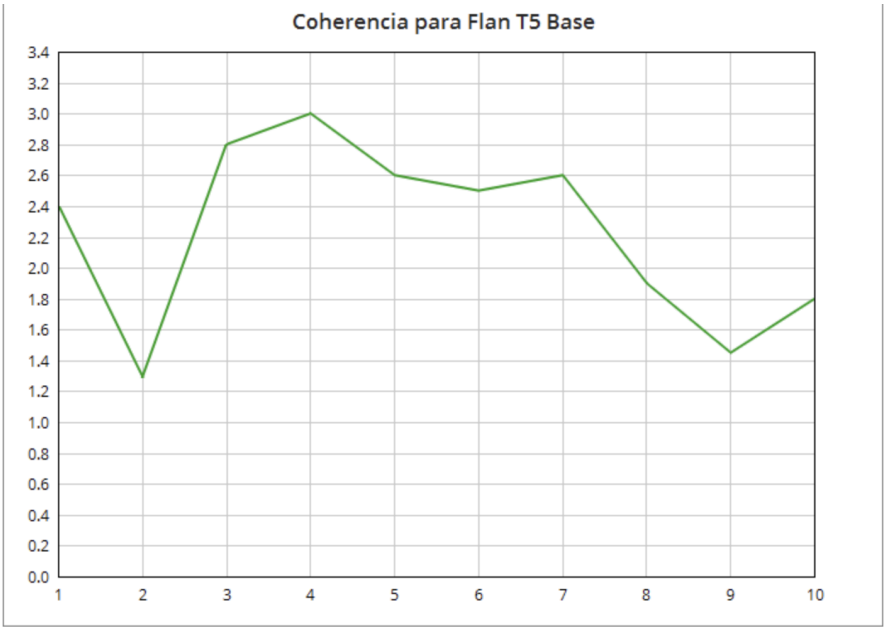


Gráfico Coherencia para Flan

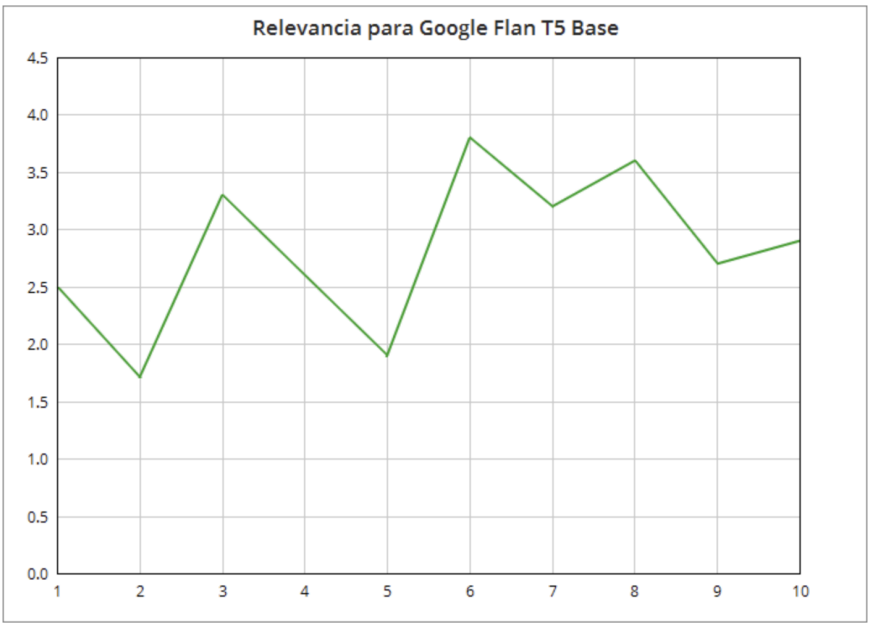


Gráfico Relevancia Flan T5 Base

9.4.3 Mistral 7B Instruct

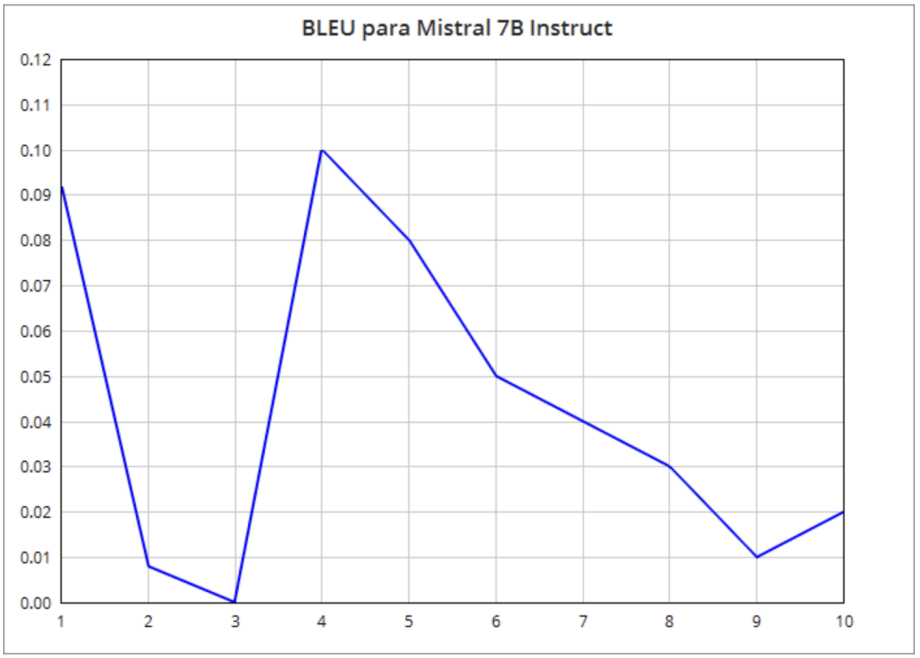


Gráfico BLEU para Mistral

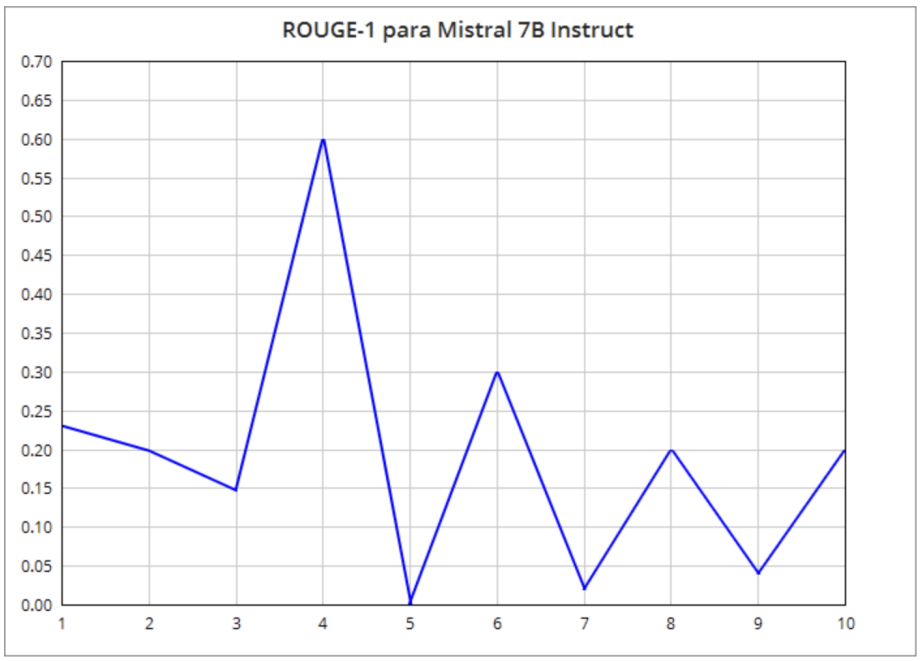


Gráfico ROUGE-1 para Mistral

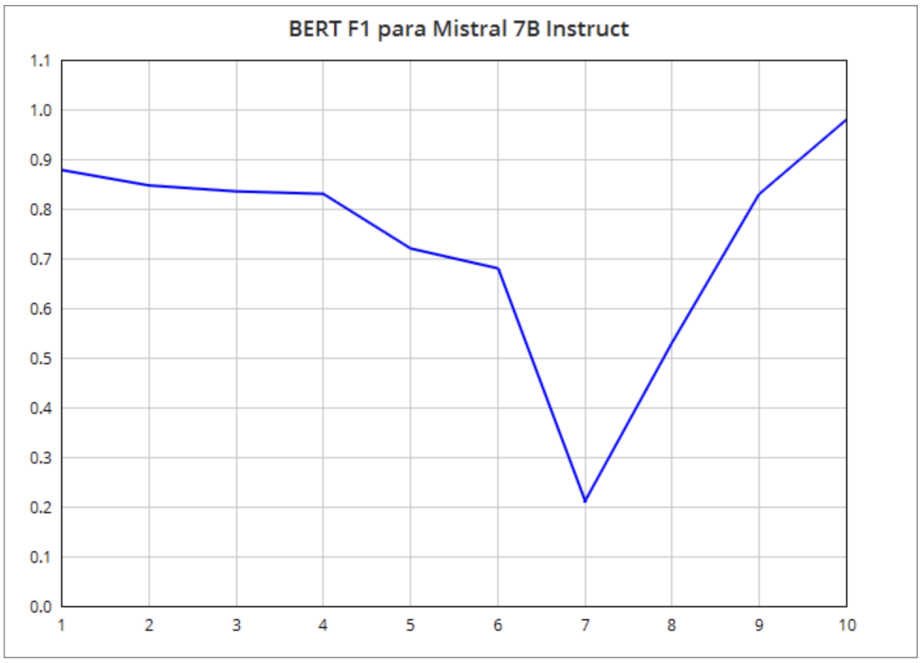


Gráfico BERT F1 para Mistral

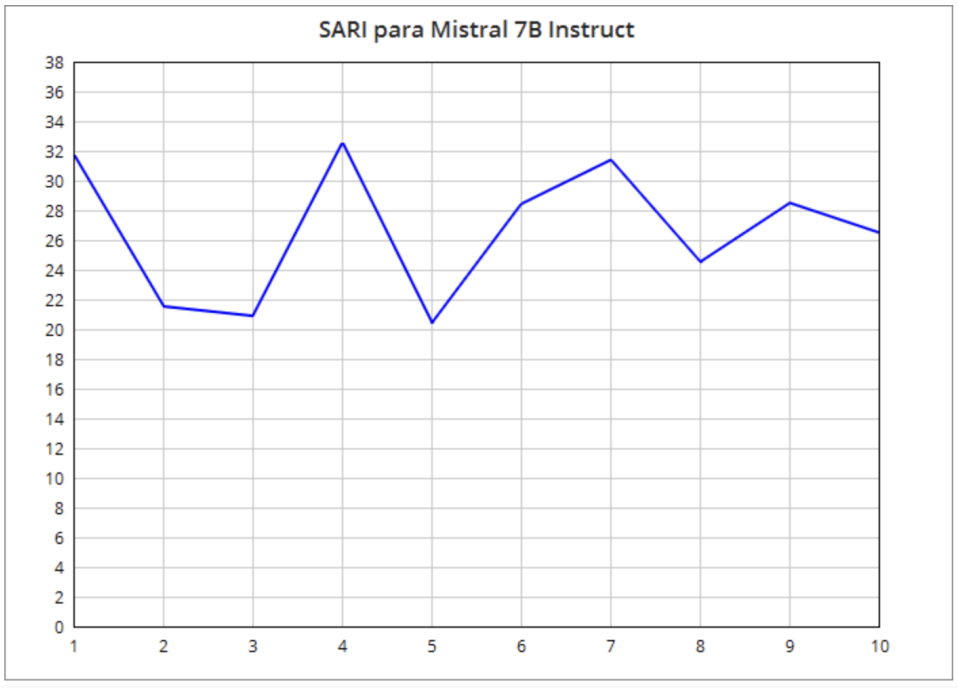


Gráfico SARI para Mistral

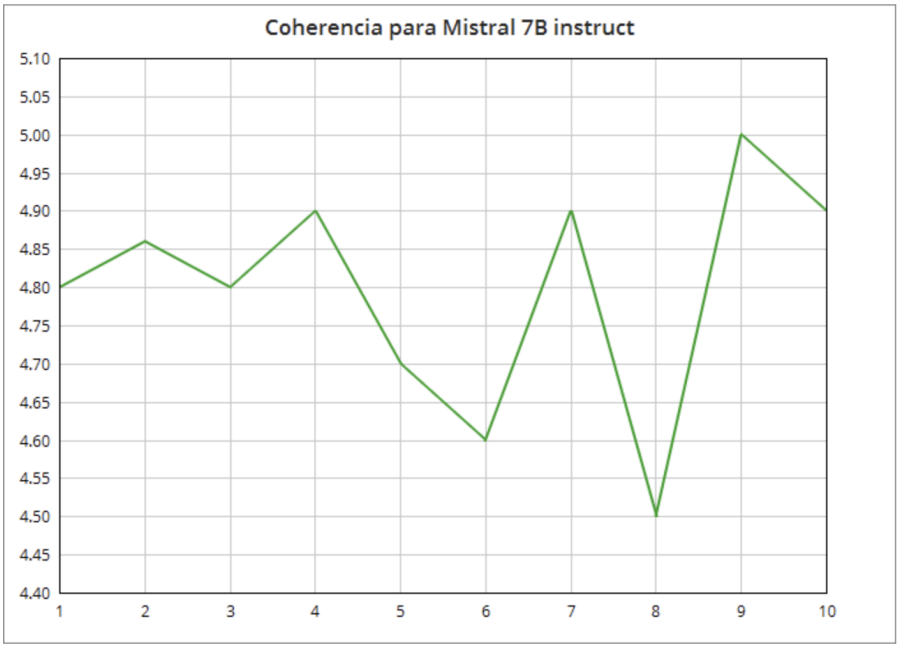


Gráfico Coherencia para Mistral

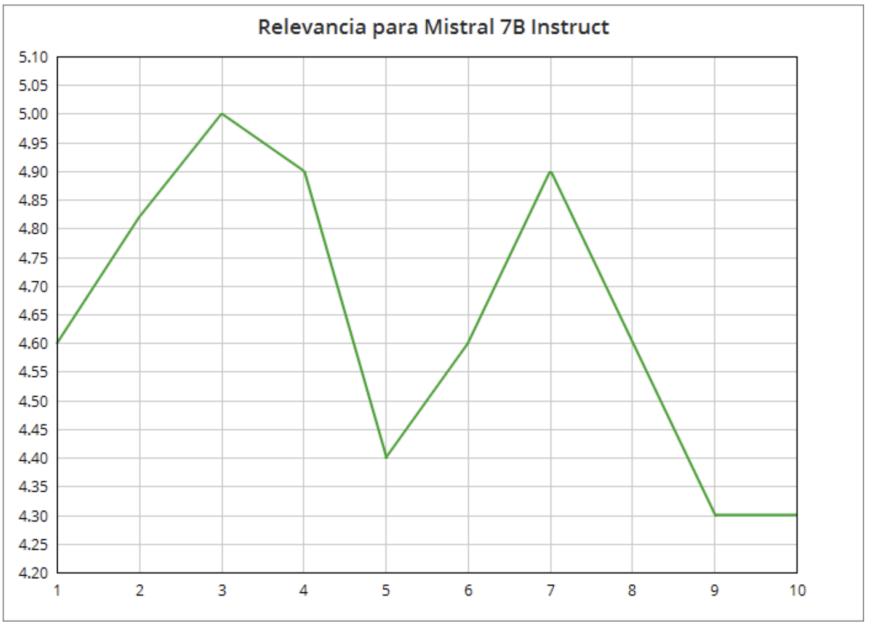


Gráfico Relevancia Mistral 7B Instruct

10 Análisis e interpretación de resultados

Para analizar los resultados obtenidos, primero se debe recordar el significado de cada valor:

Bleu

- Bleu: Se refiere a qué tan cercano es el texto generado respecto a la referencia ingresada por el usuario, siendo un valor entre 0 y 1.
- Brevity penalty: Es la penalización que se genera para poder llegar al valor de BLEU.
- Precisions: Es la media geométrica de la precisión de n-gramas, la cual asegura que la evaluación penalice fuertemente las traducciones que no logran capturar coincidencias en cualquier nivel de n-grama, promoviendo así traducciones que sean consistentes y precisas en múltiples niveles de detalle.
- Reference length: Es el número de palabras ingresadas por el usuario en la referencia.
- Result length: El número de palabras generadas por el modelo en la respuesta.

Rouge

- Rouge 1: Es la medición de 1-gramas de la respuesta generada por el modelo respecto al texto de referencia.
- Rouge 2: Es la medición de 2-gramas de la respuesta generada por el modelo respecto al texto de referencia.
- Rouge L: Mide la calidad del resumen considerando la Longitud de la Subsecuencia Común Más Larga entre el resumen generado y el resumen de referencia.
- Rouge L Sum: Variante de ROUGE-L diseñada para evaluar resúmenes de múltiples oraciones o documentos enteros. En lugar de calcular la LCS para cada oración individualmente, ROUGE-Lsum considera la LCS a nivel de todo el documento o el resumen.

BertScore

- F1: Indica que el texto generado no solo es preciso sino también completo en relación con el texto de referencia, semánticamente hablando.
- Precision: Mide la exactitud del texto generado en relación con el texto de referencia.
- Recall: Mide la completitud del texto generado en relación con el texto de referencia.

Wiki Split

- Exact: Proporciona una medida sencilla pero eficaz de cuántas oraciones generadas coinciden exactamente con las de referencia.
- Sacrebleu: evalúa la precisión de los n-gramas y aplica una penalización por brevedad, ofreciendo una medida robusta y reproducible de la calidad del texto generado.
- Sari: Mide qué tanto el modelo simplifica el texto generado respecto a la referencia.

Habiendo recordado brevemente el valor de cada métrica cualitativa y junto a las métricas cuantitativas, podemos dar un análisis general de cada modelo.

10.1 GPT 3.5 Turbo

Para entender los resultados primero hay que recordar cómo está construido y entrenado el modelo.

1. Arquitectura de Transformador (Transformer)

GPT-3.5 Turbo utiliza la arquitectura de transformador, que es eficaz para tareas de procesamiento de lenguaje natural debido a su capacidad para capturar dependencias a largo plazo en el texto. Los transformadores usan mecanismos de autoatención que permiten al modelo considerar el contexto de todas las palabras en una oración simultáneamente.

Se utiliza la técnica de autoatención, la cual permite que el modelo asigne diferentes pesos a diferentes palabras en una oración, ayudándolo a enfocarse en las partes más relevantes del texto.

2. Entrenamiento Masivo con Datos Diversos

GPT-3.5 Turbo se entrena en una vasta cantidad de datos de texto que incluye libros, artículos, sitios web, y otros textos diversos. Esta amplitud y diversidad en los datos de entrenamiento permiten al modelo manejar una amplia gama de temas y estilos de escritura.

La diversidad de los datos también introduce sesgos y potencial para alucinaciones, donde el modelo puede generar información que parece plausible pero es incorrecta o inventada. Este es un desafío conocido en los modelos grandes de lenguaje.

3. Capacidad de Parámetros

GPT-3.5 Turbo tiene miles de millones de parámetros (exactamente 175 mil millones en GPT-3), que son básicamente los pesos y sesgos ajustados durante el entrenamiento. Estos parámetros permiten al modelo captar complejidades del lenguaje humano, pero también hacen que el modelo sea computacionalmente intensivo.

Aunque un modelo grande puede generar texto coherente y relevante, también puede ser propenso a generar respuestas que no coinciden exactamente con las referencias debido a la creatividad y variabilidad inherentes en su generación de texto.

4. Ajuste Fino (Fine-Tuning)

GPT-3.5 Turbo ha sido ajustado específicamente para seguir instrucciones y participar en conversaciones, lo cual optimiza su rendimiento para tareas interactivas y de generación de texto libre. Sin embargo, este ajuste puede resultar en respuestas más variadas y creativas, en lugar de estrictamente alineadas con respuestas de referencia.

El enfoque en generación de texto conversacional puede priorizar el recall sobre la precisión, generando más contenido relevante (alto recall) a costa de incluir información adicional que puede no ser estrictamente necesaria (menor precisión).

5. Técnicas de Decodificación

La técnica de decodificación utilizada puede afectar los resultados. Por ejemplo, greedy decoding selecciona la palabra más probable en cada paso, mientras que beam search considera múltiples

posibilidades a cada paso. GPT-3.5 Turbo podría usar técnicas avanzadas de decodificación para balancear entre generar texto relevante y coherente.

Con estas bases ahora se pueden analizar los resultados obtenidos.

10.1.1 Precisión y coincidencia de palabras (BLEU y SacreBLEU)

- BLEU: 0.069
- Brevity Penalty: 1.0
- BLEU Precision 1-gram: 0.191
- BLEU Precision 2-gram: 0.134
- BLEU Precision 3-gram: 0.091
- BLEU Precision 4-gram: 0.033
- SacreBLEU: 9.812

El BLEU Score es bastante bajo, lo que indica que las respuestas generadas por GPT-3.5 Turbo tienen una coincidencia casi nula con las referencias. Esto sugiere que el modelo puede estar generando texto que no se alinea estrechamente con las respuestas esperadas, lo cual no es necesariamente malo, ya que hay que recordar que las referencias son palabras esperadas por cada usuario, y no necesariamente la no coincidencia significa un error. Esto más bien se debe a la creatividad y capacidad de generar respuestas diferentes a lo esperado en cuanto a formulación.

Aunque GPT-3.5 Turbo es un modelo grande y potente, no siempre prioriza la coincidencia exacta de palabras, ya que su enfoque está más en la comprensión semántica y la generación coherente.
El modelo tiende a generar respuestas variadas, lo cual puede reducir la precisión n-grama. Esto refleja su capacidad para generar un lenguaje natural fluido, pero puede sacrificar la precisión textual.

La arquitectura de transformers se enfoca en el contexto global, lo cual puede llevar a una reformulación de frases que, aunque semánticamente correctas, no coinciden exactamente con las referencias, que hay que recordar que son generadas por el usuario, por lo que varía en la estructura que haya hecho cada uno.
Como un modelo autoregresivo, GPT-3.5 Turbo genera cada token uno a uno basándose en los tokens anteriores. Esto puede llevar a variaciones en la formulación de frases, lo que reduce la precisión de n-gramas cuando se compara con el texto de referencia.

Con 175 mil millones de parámetros, el modelo tiene una capacidad inmensa para generar texto variado y coherente. Sin embargo, esta misma capacidad introduce una alta variabilidad, lo que puede reducir los puntajes de BLEU y SacreBLEU cuando se requiere una coincidencia exacta.

10.1.2 ROUGE

- ROUGE-1: 0.315
- ROUGE-2: 0.221
- ROUGE-L: 0.313
- ROUGE-Lsum: 0.313

Los resultados de ROUGE indican una capacidad moderada para capturar información relevante y estructura textual. El modelo es eficaz en comprender y capturar palabras y frases clave debido a su capacidad para procesar grandes contextos. Aunque las secuencias exactas de palabras pueden variar, la estructura general y el contenido relevante suelen estar presentes, lo que se refleja en los puntajes de ROUGE. Sin embargo, la capacidad de capturar secuencias más largas de manera precisa es limitada, lo que se observa en los puntajes más bajos de ROUGE-2 y ROUGE-L.

GPT-3.5 Turbo ha sido pre entrenado en un corpus extenso y diverso. Esto mejora su capacidad para comprender y replicar estructuras textuales, lo que resulta en buenos puntajes de ROUGE al capturar información relevante y estructurada.

La capacidad del modelo para enfocarse en múltiples partes del texto simultáneamente (gracias a la atención multicabezal) le permite capturar palabras y frases clave, mejorando los puntajes de ROUGE. El modelo maneja bien secuencias largas, lo que le permite mantener la coherencia y capturar la estructura del texto a lo largo de los párrafos.

Este métrica reafirma lo visto con el puntaje BLEU, las coincidencias de texto no son exactas, pero las pocas que llega a haber normalmente son las partes más relevantes de las referencias, lo que reafirma su capacidad de decir información precisa pero con diversas formas de plantear las oraciones.

10.1.3 BERTScore

- BERTScore F1: 0.86
- BERTScore Precision: 0.84
- BERTScore Recall: 0.88

El alto puntaje en BERTScore demuestra la capacidad del modelo para generar respuestas que son semánticamente similares a las referencias, mostrando palabras clave en el tema que se está tratando.

La arquitectura de GPT-3.5 Turbo permite una comprensión profunda del lenguaje y el contexto, lo que facilita la generación de texto semánticamente similar, dicha arquitectura facilita la creación de representaciones contextuales ricas para cada token, lo que mejora la similitud semántica entre las

respuestas generadas y las referencias. De igual forma, el pre entrenamiento en grandes cantidades de texto diverso ayuda al modelo a capturar y replicar relaciones semánticas complejas..

La alta precisión y recall indican que el modelo no solo genera respuestas que son correctas, sino que también incluye la mayor parte de la información relevante de las referencias. Esto no se contradice con las puntuaciones anteriores, más bien, el hecho de coincidir semánticamente nos habla de la precisión al tocar los temas clave de la pregunta realizada.

10.1.4 WikiSplit

- SARI: 9.812
- Exact match: 0

El valor bajo de SARI habla de que los textos generados no suelen estar tan resumidos, por lo que se intuye que muchas veces dan más texto del necesario además del tema principal.

La baja coincidencia exacta indica cómo el modelo genera respuestas que no coinciden en lo absoluto con las frases de referencia, mostrando creatividad para dar información.

Esto podría tener como causas que GPT-3.5 Turbo está diseñado para ser flexible y creativo en la generación de texto, lo que puede resultar en respuestas que no coinciden exactamente con las frases de referencia. La configuración de millones de parámetros permite variabilidad en las respuestas es beneficioso para tareas de generación de lenguaje natural, pero no para coincidencia exacta.

Además el modelo está más orientado a mantener la coherencia y relevancia semántica en lugar de la coincidencia exacta de palabras.

10.1.5 Evaluación cualitativa

- Coherencia: 4.69
- Relevancia: 4.623
- Fluidez: 4.88
- Creatividad: 3.83

Los puntajes cualitativos reflejan un rendimiento sólido en la mayoría de las áreas:

- Coherencia: La alta puntuación sugiere que el modelo mantiene una narrativa lógica y consistente, gracias a su capacidad para procesar y generar texto basado en contextos amplios.

- Relevancia: La puntuación alta en relevancia indica que el modelo entiende bien las preguntas y proporciona información relevante, reflejando su capacidad de pre entrenamiento en grandes cantidades de datos.
- Fluidez: La alta fluidez es esperada en un modelo de gran tamaño como GPT-3.5 Turbo, que está diseñado para generar texto que fluye de manera natural.
- Creatividad: La puntuación más baja en creatividad puede ser debido a que el modelo tiende a generar respuestas precisas y coherentes, lo que puede limitar la variabilidad y originalidad en algunas respuestas.

10.1.6 Conclusiones

GPT-3.5 Turbo muestra un rendimiento sólido en términos de coherencia, relevancia y fluidez, reflejando su capacidad para generar texto semánticamente rico y bien estructurado. Sin embargo, sus puntajes en precisión n-gram y coincidencia exacta indican que tiende a reformular el texto en lugar de seguir secuencias exactas de palabras. Esta característica es tanto una fortaleza, en términos de generar lenguaje natural y coherente, como una limitación para tareas que requieren coincidencia precisa con el texto de referencia.

El análisis sugiere que GPT-3.5 Turbo es ideal para aplicaciones donde la comprensión semántica y la generación fluida de texto son cruciales, pero puede no ser la mejor opción para tareas que requieren precisión textual exacta. La arquitectura y el pre entrenamiento extenso del modelo lo hacen apto para capturar el contexto y la relevancia, aunque podría beneficiarse de ajustes adicionales para mejorar en áreas de reformulación precisa y creatividad.

10.2 Google Flan T5 Base

Los puntos clave en la arquitectura de este modelo son:

- Transformador: Flan-T5 utiliza la arquitectura de transformador, que es eficiente para el aprendizaje del contexto y la generación de texto. La técnica de atención que emplea permite al modelo entender la relación entre las palabras en una oración y generar texto coherente.
- Entrenamiento Previo: Está entrenado en una variedad de tareas de comprensión y generación de texto, lo que lo hace versátil para múltiples aplicaciones. Sin embargo, esta versatilidad puede resultar en respuestas que no siempre se alinean perfectamente con una referencia específica.

Analizando los resultados obtenidos se llega a lo siguiente:

10.2.1 BLEU

- BLEU: 0.209
- Brevity Penalty: 0.626

- BLEU Precision 1-gram: 0.416
- BLEU Precision 2-gram: 0.318
- BLEU Precision 3-gram: 0.184
- BLEU Precision 4-gram: 0.144
- SacreBLEU: 21.14

El puntaje BLEU de 0.209 indica que las respuestas generadas por Google Flan T5 Base tienen una moderada coincidencia con las respuestas de referencia, es decir, no siempre utiliza las mismas palabras exactas o frases que se esperaban. Aún así, es ligeramente más algo que GPT, lo que puede significar que tiene más coincidencia con las referencias.

El Brevity Penalty de 0.626 sugiere que las respuestas del modelo tienden a ser más largas o más cortas que las respuestas de referencia, lo que penaliza el puntaje BLEU final.

Las precisiones en los n-gramas (combinaciones de 1, 2, 3, o 4 palabras) muestran que el modelo es relativamente bueno en coincidir con palabras individuales y pares de palabras, pero su precisión disminuye con combinaciones más largas, lo que sugiere que el modelo puede perder precisión en frases más complejas. Flan-T5 Base está basado en la arquitectura T5, que también utiliza mecanismos de atención para procesar y generar texto. Sin embargo, su pre entrenamiento y afinación específica podrían estar optimizados para tareas que no necesariamente se alinean con las coincidencias exactas de n-gramas.

El T5 fue diseñado originalmente para tareas de transformación de texto (como traducción, resumen, etc.), lo que le da una ventaja en la reformulación y precisión de palabras a nivel superficial.

10.2.2 ROUGE

- ROUGE-1: 0.414
- ROUGE-2: 0.301
- ROUGE-L: 0.414
- ROUGE-Lsum: 0.414

Los puntajes ROUGE-1 y ROUGE-2 indican que el modelo captura una buena cantidad de información relevante, ya que hay una considerable coincidencia entre las palabras individuales y las combinaciones de dos palabras en las respuestas generadas y las de referencia. Google Flan T5 Base ha sido pre entrenado en un amplio conjunto de datos multilingües y multitareas, lo que le permite capturar información relevante de manera efectiva.

El puntaje ROUGE-L, que mide las coincidencias de las subsecuencias más largas, muestra que el modelo mantiene la estructura y coherencia general del contenido esperado. La capacidad del

modelo para manejar el contexto y las relaciones entre palabras y frases mejora sus puntajes en ROUGE al capturar la información esencial y mantener la estructura del texto.

10.2.3 BERTScore

- BERTScore F1: 0.87
- BERTScore Precision: 0.88
- BERTScore Recall: 0.41

El alto puntaje de precisión (0.88) y F1 (0.87) refleja la capacidad del modelo para generar respuestas que son semánticamente similares a las referencias. El T5, al igual que otros modelos de transformadores, crea representaciones ricas en contexto que ayudan a capturar la similitud semántica entre el texto generado y el de referencia.

Los puntajes altos en precisión y F1 indican que las respuestas del modelo son semánticamente similares a las de referencia, capturando bien el significado general. Sin embargo, el puntaje bajo en recall sugiere que el modelo puede dejar de lado algunos detalles importantes.

10.2.4 WikiSplit

- WikiSplit Exact: 0.13
- WikiSplit SacreBLEU: 21.14

Al igual que con otros modelos de generación de lenguaje, Flan-T5 Base tiende a ser más flexible y creativo en la generación de texto, lo que puede afectar negativamente su puntaje de coincidencia exacta.

Los puntajes relativamente bajos en la coincidencia exacta indican que el modelo puede no estar completamente optimizado para tareas de reformulación precisa, enfocándose más en la transformación y la adaptación del texto.

10.2.5 Evaluación cualitativa

- Coherencia: 2.27
- Relevancia: 2.85
- Fluidez: 1.98
- Creatividad: 1.52

Como se puede ver, aunque en las métricas cuantitativas los valores no fueron malos, de forma cuantitativa los resultados dejaron mucho que desear. Especialmente, se notó la forma en la que el modelo suele capturar y buscar la información tal cual la tiene, sin dar más contexto, pero cuando no

encuentra la respuesta o no entiende la pregunta planteada, suele responder cosas con poco sentido.

El puntaje de 2.27 en coherencia sugiere que las respuestas generadas pueden ser comprensibles pero no siempre son completamente coherentes en el contexto.

Con un puntaje de 2.85 en relevancia, el modelo a menudo proporciona información útil, pero puede incluir datos irrelevantes o no enfocarse completamente en la pregunta.

El bajo puntaje en fluidez (1.98) indica que las respuestas pueden no ser tan naturales o fáciles de leer, quizás debido a una estructura de oración incómoda o errores gramaticales.

El puntaje de 1.52 en creatividad sugiere que el modelo tiende a ser bastante literal y no proporciona respuestas particularmente originales o innovadoras.

Los puntajes bajos en coherencia, relevancia, fluidez y creatividad pueden deberse a que Flan-T5 Base no está específicamente ajustado para la generación de texto de alta calidad en tareas conversacionales, sino más bien para una variedad de tareas de NLP.

A pesar de ser un modelo robusto, la generalización puede llevar a respuestas menos enfocadas y menos naturales en ciertos contextos, afectando la calidad percibida de las respuestas.

La arquitectura T5, aunque poderosa, puede tener dificultades para adaptar y generar texto altamente natural y coherente en contextos abiertos, lo que se refleja en las puntuaciones cualitativas.

10.3.6 Conclusiones

Google Flan T5 Base muestra un desempeño moderado en métricas de precisión y coincidencia de palabras debido a su enfoque en tareas de transformación de texto y su arquitectura basada en transformadores. Sin embargo, su capacidad para generar respuestas semánticamente similares es fuerte, como lo indican los altos puntajes de BERTScore en precisión y F1. Los resultados en WikiSplit y las métricas cualitativas sugieren que el modelo puede no estar completamente optimizado para generar texto altamente natural y coherente en tareas abiertas, lo que afecta su relevancia, fluidez y creatividad. Estas características reflejan las fortalezas y limitaciones del modelo en función de su arquitectura y entrenamiento.

10.3 MistralAI Mistral-7B-Instruct-v0.2

Como modelo basado en la arquitectura de transformador, Mistral-7B está optimizado para capturar el contexto en el texto y generar respuestas coherentes. Sin embargo, su rendimiento indica que

puede necesitar ajustes adicionales para mejorar la precisión y la coherencia con las respuestas de referencia.

Mistral-7B está ajustado para tareas instructivas, lo que podría llevar a respuestas que son más variadas y menos estructuradas en comparación con las respuestas esperadas. Esta flexibilidad puede ser útil en aplicaciones donde la creatividad es valorada más que la precisión exacta.

10.3.1 BLEU

- BLEU: 0.043
- Brevity Penalty: 1
- BLEU Precision 1-gram: 0.113
- BLEU Precision 2-gram: 0.057
- BLEU Precision 3-gram: 0.037
- BLEU Precision 4-gram: 0.026
- SacreBLEU: 5.55

Los puntajes de BLEU y Precisión de n-gramas son bastante bajos, lo que significa que el modelo no está coincidiendo bien con las respuestas de referencia en términos de las secuencias exactas de palabras. En otras palabras, las frases generadas por el modelo no son muy similares a las respuestas correctas esperadas. Similar a BLEU, el puntaje de SacreBLEU es bajo, sugiriendo que el modelo no produce textos que coinciden bien con las respuestas esperadas. Pero como se mencionó en los anteriores análisis, esto no necesariamente significa que el modelo sea malo.

La penalización por brevedad (1) indica que las respuestas generadas tienen una longitud adecuada, lo cual es positivo y muestra que el modelo no está produciendo respuestas excesivamente cortas o largas.

Los valores decrecientes en las precisiones de n-gramas (1-grama a 4-grama) reflejan una capacidad limitada del modelo para generar secuencias de palabras exactas que coincidan con las respuestas de referencia. Esto puede sugerir que el modelo tiene una estructura de entrenamiento que no prioriza las coincidencias exactas de palabras.

10.3.2 ROUGE

- ROUGE-1: 0.180
- ROUGE-2: 0.358
- ROUGE-L: 0.126
- ROUGE-Lsum: 0.180

Estos puntajes indican que el modelo capta algunas palabras y pares de palabras importantes de las respuestas correctas, pero no en gran medida. Significa que el modelo puede extraer cierta información relevante, pero no lo suficiente como para ser altamente efectivo.

Los puntajes bajos sugieren que el modelo no sigue bien la estructura y secuencia de las respuestas correctas. Esto implica que, aunque las respuestas pueden ser coherentes, no siempre mantienen la misma estructura que las respuestas esperadas. Una vez más, hablando de la capacidad de formular oraciones en con distintas estructuras.

10.3.3 BERTScore

- BERTScore F1: 0.737
- BERTScore Precision: 0.853
- BERTScore Recall: 0.88

Estos puntajes son relativamente altos, lo que significa que las respuestas generadas por el modelo son semánticamente similares a las respuestas correctas. El modelo entiende bien el significado subyacente de las preguntas y respuestas, incluso si no coincide exactamente en las palabras utilizadas.

El puntaje F1 de 0.737 refleja una combinación sólida de precisión y recall, aunque hay margen de mejora.

10.3.4 WikiSplit

- WikiSplit Exact: 0
- WikiSplit SacreBLEU: 5.55

El puntaje exacto de 0 indica que el modelo no produce respuestas que coincidan exactamente con las de referencia. Esto puede deberse a la flexibilidad del modelo en generar texto y a la falta de enfoque en coincidencias exactas durante el entrenamiento.

Un puntaje de SacreBLEU de 5.55 muestra que, aunque hay cierta similitud en términos de contenido, el modelo no está optimizado para coincidir exactamente con las frases de referencia. Esto podría ser indicativo de un modelo entrenado para una generación de texto más creativa y flexible.

10.3.5 Evaluación cualitativa

- Coherencia: 4.8

- Relevancia: 4.7
- Fluidez: 4.7
- Creatividad: 4.4

La alta coherencia indica que el modelo produce respuestas lógicas y bien organizadas. Las respuestas tienen sentido y siguen un flujo natural, lo que es crucial para mantener una conversación clara y comprensible. Esto sugiere una arquitectura que maneja bien la continuidad y las relaciones contextuales dentro del texto.

Un puntaje alto en relevancia muestra que el modelo es bueno para proporcionar información pertinente y útil. Responde directamente a las preguntas planteadas y ofrece datos relevantes.

La fluidez alta significa que las respuestas son gramaticalmente correctas y fáciles de leer. Esto sugiere que el modelo ha sido entrenado en un corpus de texto de alta calidad que incluye muchas formas naturales de expresión.

El modelo muestra un buen nivel de creatividad, generando respuestas innovadoras y originales. Esto es positivo para aplicaciones que valoran respuestas únicas y detalladas, aunque puede afectar la precisión cuando se requiere una coincidencia exacta.

10.3.6 Conclusiones

Mistral-7B-Instruct-v0.2 demuestra ser un modelo robusto en términos de coherencia, relevancia y fluidez, lo que refleja una arquitectura bien diseñada para la generación de texto de alta calidad. Sin embargo, sus puntajes más bajos en las métricas BLEU, ROUGE y WikiSplit indican que hay margen para mejorar en la precisión y la coincidencia exacta de palabras y frases con las respuestas de referencia. Las evaluaciones cualitativas altas sugieren que el modelo es adecuado para aplicaciones donde la naturalidad y la relevancia son más importantes que la coincidencia exacta de texto. La flexibilidad y creatividad del modelo son aspectos destacados, aunque a costa de la precisión en la generación de respuestas predecibles y estructuradas.

10.4 Análisis general

Precisión de Coincidencia (BLEU y SacreBLEU)

- GPT-3.5 Turbo presenta una puntuación de BLEU baja, lo que sugiere una coincidencia limitada con las secuencias de referencia. Su SacreBLEU es notablemente superior a los de Google Flan-T5 Base y Mistral-7B, aunque aún no es ideal. Esto indica una cierta capacidad para generar respuestas similares en términos de longitud y contenido, pero con una considerable variabilidad.

- Google Flan-T5 Base destaca con una puntuación de BLEU significativamente más alta (0.3) y una Brevity Penalty de 0.9, lo que indica una buena aproximación en longitud a las

respuestas de referencia y una mejor capacidad para capturar la estructura n-gram en comparación con los otros modelos. Sin embargo, su SacreBLEU sugiere que aunque el modelo se aproxima a la estructura deseada, aún tiene espacio para mejorar en precisión general.

- Mistral-7B-Instruct-v0.2 tiene las puntuaciones de BLEU y SacreBLEU más bajas. Esto indica una gran variabilidad y poca coincidencia exacta con las secuencias de referencia, lo que podría ser reflejo de su enfoque en la generación de texto variado y menos estructurado.

Relevancia y Captura de Palabras Clave (ROUGE)

- GPT-3.5 Turbo tiene puntuaciones de ROUGE moderadas, lo que sugiere una capacidad razonable para capturar palabras y estructuras clave en las respuestas, aunque con margen para mejorar en la coincidencia de secuencias más largas.

- Google Flan-T5 Base muestra fuertes resultados en todas las variantes de ROUGE, lo que indica una mejor capacidad para capturar palabras clave y estructuras de oración relevantes en comparación con los otros modelos. Esto sugiere que Flan-T5 genera texto que es más relevante y alineado con las expectativas de las respuestas de referencia.

- Mistral-7B-Instruct-v0.2 tiene puntuaciones de ROUGE bajas, lo que indica una menor capacidad para capturar palabras clave y la estructura en las respuestas generadas. Esto refuerza la observación de que el modelo tiende a generar respuestas menos alineadas con las expectativas de las referencias.

Similitud Semántica (BERTScore)

- GPT-3.5 Turbo sobresale en la métrica BERTScore, lo que sugiere que captura de manera efectiva la similitud semántica y el significado general del texto de referencia, a pesar de no coincidir textualmente.
- Google Flan-T5 Base también muestra un fuerte rendimiento en BERTScore. Esto indica que sus respuestas no solo coinciden bien en términos de estructura y longitud, sino que también mantienen un alto grado de similitud semántica con el texto de referencia.

- Mistral-7B-Instruct-v0.2 tiene un BERTScore más bajo en comparación con los otros dos modelos, lo que sugiere una menor capacidad para mantener la similitud semántica con las respuestas de referencia, aunque aún dentro de un rango aceptable.

Coincidencia Exacta y Reformulación (Exact Match y SARI)

- GPT-3.5 Turbo muestra una baja coincidencia exacta (0), lo que indica que ninguna de sus respuestas coincide textualmente con las de referencia. Su puntuación SARI sugiere que es moderadamente eficaz en la reformulación de frases, capturando algún grado de precisión y relevancia en la generación de respuestas.

- Google Flan-T5 Base tiene una puntuación de Exact Match de 0.2, indicando que una proporción pequeña pero significativa de sus respuestas coinciden textualmente con las de referencia. Su puntuación SARI es más baja que la de GPT-3.5 Turbo, lo que sugiere una menor efectividad en la reformulación de frases, aunque aún es competente en la generación de respuestas precisas.

- Mistral-7B-Instruct-v0.2 también tiene una puntuación de Exact Match de 0, reflejando la alta variabilidad en sus respuestas. Su puntuación SARI es similar a la de Google Flan-T5 Base, indicando una capacidad moderada para reformular frases de manera precisa y relevante.

1. GPT-3.5 Turbo
    - Arquitectura: Basado en la arquitectura Transformer, entrenado en un vasto corpus de datos con 175 mil millones de parámetros. Esta robustez se traduce en una capacidad superior para capturar la similitud semántica, aunque la precisión textual puede ser limitada debido a la tendencia del modelo a generar respuestas variadas.

    - Fuerzas: Alta similitud semántica y relevancia general. Buen manejo de la longitud y estructura en las respuestas.

    - Áreas de Mejora: Precisión textual y coincidencia exacta con las respuestas de referencia. Mayor alineación con la estructura esperada de las respuestas.

2. Google Flan-T5 Base
    - Arquitectura: Basado en T5, optimizado para tareas de generación y comprensión del lenguaje. Su capacidad para comprender y generar texto con alta precisión y relevancia es notable, aunque con un tamaño de modelo más pequeño (220 millones de parámetros en la versión Base).

    - Fuerzas: Alta precisión n-gram, relevancia y captura de la estructura del texto. Buena similitud semántica y coincidencia textual.

    - Áreas de Mejora: Reformulación de frases (SARI), y en algunos casos, la longitud y coincidencia exacta con las respuestas de referencia.

3. Mistral-7B-Instruct-v0.2

- Arquitectura: Modelo de 7 mil millones de parámetros, diseñado para generar respuestas instructivas. Su enfoque está en proporcionar variedad y flexibilidad en las respuestas, lo que puede sacrificar precisión y coincidencia con el texto de referencia.

- Fuerzas: Generación de respuestas semánticamente relevantes. Buena capacidad para manejar contextos variados.

- Áreas de Mejora: Coincidencia n-gram y estructural (BLEU y ROUGE). Mayor precisión y relevancia en la captura de palabras clave y estructura de las respuestas.

Cada uno de estos modelos muestra fortalezas y debilidades específicas según las métricas evaluadas:

- GPT-3.5 Turbo es fuerte en similitud semántica y relevancia general, adecuado para aplicaciones donde la comprensión y generación de significado profundo es crucial.

- Google Flan-T5 Base sobresale en precisión n-gram y relevancia textual, siendo ideal para tareas que requieren coincidencia precisa con el texto de referencia.

- Mistral-7B-Instruct-v0.2 proporciona flexibilidad y diversidad en la generación de respuestas, útil en escenarios que valoran la creatividad y variabilidad en la respuesta.

En cuanto a la evaluación cualitativa, GPT y Mistral se asemejan bastante, teniendo resultados muy altos, en cuanto a la percepción del usuario hacia ellos. Normalmente sus respuestas son coherentes y fluidas, lo que los hace ideales para mantener conversaciones. Flan si bien tiene resultados bajos en la percepción del usuario, esto se debió principalmente a que este modelo suele devolver la información exacta como respuesta, lo que lo hace ideal para otro tipo de tareas que no sean necesariamente chats con usuarios humanos.

Se notó una gran similitud tanto en resultados cuantitativos como cualitativos y la percepción del usuario en los modelos de GPT y Mistral, aunque la cantidad de parámetros que poseen sean de una gran diferencia. Se intuye que esta similitud entre los dos modelos y su diferencia con Flan, se deben principalmente a su arquitectura, mientras que Flan está hecho con una arquitectura T5, GPT y Mistral están hechos con transformers. Si bien T5 está basado en transformers, este suele contar con un enfoque unificado y optimizaciones específicas para transformar todas las tareas de procesamiento de lenguaje natural en problemas de texto a texto, lo que proporciona flexibilidad y eficiencia en una amplia gama de aplicaciones de NLP. Los Transformers generales, por otro lado, son más versátiles y pueden ser adaptados a diversas tareas, pero pueden no tener las mismas ventajas en términos de enfoque unificado y eficiencia para múltiples tareas simultáneas.

La similitud entre GPT y Mistral hace que la diferencia en el número de parámetros (200 mil millones y 7 mil millones respectivamente) parezca despreciable, al menos para los efectos de un chatbot como el que se trabajó. Claramente, tener más parámetros hará que en ciertos puntos o con usos más complejos se note la diferencia. Sin embargo, para un usuario común, al menos en las pruebas realizadas, no representó grandes cambios. En cambio, Flan (220 millones de parámetros) tuvo problemas, especialmente al formular lenguaje, ya que muchas veces replicaba la información tal cual estaba representada en el texto proporcionado. Por lo tanto, se puede decir que, dentro de cierto rango, la cantidad de parámetros puede ser indistinta para efectos de crear un modelo que genere lenguaje lo más elaborado y coherente posible, siempre y cuando se llegue a un número suficiente de parámetros.

Al final de cuentas, las pruebas expresan la forma en la que cada modelo puede ser distinto y útil para diferentes cosas dependiendo de factores puntuales como lo son los corpus de datos con los que se entrenaron, su arquitectura y el número de parámetros con los que fueron hechos. Estos tres factores son puntuales al momento de crear un modelo, y se deben elegir cuidadosamente dependiendo de la finalidad que se le quiera dar.

11 Conclusiones

En el pasado, la gente solía imaginar la década de los 2000 con autos voladores y teletransportación de personas a cualquier lugar. Es difícil saber si la humanidad está cerca de lograr algo así (o si ya lo ha logrado). Este trabajo de tesina planteó el objetivo de aportar un poco de conocimiento e investigación sobre el trabajo de los ingenieros en la elaboración de modelos generadores de texto. Estos modelos, hoy en día, aportan mucho a la comunidad, sirviendo como herramientas en una gran cantidad de rubros. La evaluación de los modelos no se realizó con el fin estricto de declarar un ganador (aunque cada persona y usuario que realizó las pruebas pudo tener uno en mente), sino más bien para comprender en dónde estamos en materia de Inteligencia Artificial y qué pasos debemos seguir para continuar avanzando.

En el presente trabajo, se exploraron las bases del procesamiento del lenguaje natural (NLP) y cómo funcionan los grandes ejemplos de éxito actual como GPT de OpenAI, Flan de Google y Mistral de MistralAI. No se buscó replicar exactamente su forma de trabajo o la manera en la que cada uno desarrolló su modelo. Más bien, se estableció una idea estándar para comprender por qué su trabajo está llegando a la vida de la gente en todo el mundo. De esta manera, cualquier persona interesada en conocer un poco sobre un modelo de lenguaje actual puede tener una introducción con ejemplos reales. Los resultados de las métricas proporcionaron una comprensión más profunda de qué es lo que cada modelo está enviando al usuario, permitiendo verlo desde múltiples ángulos y no solo desde la perspectiva del usuario final.

En el futuro, se espera que el conocimiento recopilado sirva de base para los próximos ingenieros que deseen incursionar en el mundo de la Inteligencia Artificial, la cual avanza a pasos agigantados y (esperamos) muy pronto hará que toda la información contenida en este documento sea totalmente obsoleta. Lo cual no sería ningún problema, ya que eso es justamente lo que se busca con esto: avanzar en el conocimiento, aprender unos de otros, transfiriendo información de los que saben hacia los que no saben, tal como lo ha hecho la humanidad a lo largo de la historia.

12 Referencias

Asale, R.-. (s. f.). Inteligencia | Diccionario de la Lengua Española. «Diccionario de la lengua española» - Edición del Tricentenario. https://dle.rae.es/inteligencia

Walther. (2023, 2 mayo). ¿Qué son los modelos de inteligencia artificial y cuáles son los más usados? Tutoriales Dongee.
https://www.dongee.com/tutoriales/que-son-los-modelos-de-inteligencia-artificial-y-cuales-son-los-mas-usados/#:~:text=Los%20modelos%20de%20inteligencia%20artificial%20(IA)%20son%20algoritmos%20y%20enfoques,en%20m%C3%A1quinas%20y%20sistemas%20inform%C3%A1ticos.

Biografia de Alfred Binet. (s. f.). https://www.biografiasyvidas.com/biografia/b/binet.htm

Prego, C. (2022, 20 febrero). La inteligencia artificial lleva años ganándonos al ajedrez. ahora también nos está enseñando a mejorarlo. Xataka.
https://www.xataka.com/inteligencia-artificial/inteligencia-artificial-no-solo-nos-gana-al-ajedrez-ahora-nos-muestra-tambien-como-mejorarlo

¿Qué son las redes neuronales? | IBM. (s. f.). https://www.ibm.com/mx-es/topics/neural-networks

¿Qué es machine learning? | IBM. (s. f.). https://www.ibm.com/mx-es/topics/machine-learning

Modelos de aprendizaje profundo preentrenados | Extracción de entidades de imágenes y mucho más. (s. f.). https://www.esri.com/es-es/arcgis/deep-learning-models

Eteimorde, Y. (2023, 16 agosto). Understanding LangChain's RecursiveCharacterTextSplitter. DEV Community. https://dev.to/eteimz/understanding-langchains-recursivecharactertextsplitter-2846

Nodd3r. (n.d.). Test de Turing. https://nodd3r.com/blog/test-de-turing

¿Qué son los modelos LLM? | Grandes modelos de lenguaje | Cloudflare. (n.d.).
https://www.cloudflare.com/es-es/learning/ai/what-is-large-language-model/

Weller, D. (2022, November 5). How to help your students by teaching chunks of language. Barefoot TEFL Teacher. https://www.barefootteflteacher.com/p/teaching-chunks-of-language

Espíndola, G. (2023, March 3). ¿Qué son los embeddings y cómo se utilizan en la inteligencia artificial con python? Medium.
https://gustavo-espindola.medium.com/qu%C3%A9-son-los-embeddings-y-c%C3%B3mo-se-utilizan-en-la-inteligencia-artificial-con-python-45b751ed86a5

colaboradores de Wikipedia. (2024, April 16). BLEU. Wikipedia, La Enciclopedia Libre.
https://es.wikipedia.org/wiki/BLEU

Srivastava, T. (2024, January 8). 12 Important model evaluation Metrics for Machine Learning Everyone should know (Updated 2023). Analytics Vidhya.
https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/

Chiusano, F. (2022, January 15). Two minutes NLP — Learn the BLEU metric by examples. Medium.
https://medium.com/nlplanet/two-minutes-nlp-learn-the-bleu-metric-by-examples-df015ca73a86

Qué es la inteligencia artificial: definición, historia, aplicaciones y futuro. (n.d.). Tableau.
https://www.tableau.com/es-mx/data-insights/ai/what-is

Chiusano, F. (2023, August 4). Two minutes NLP — Learn the ROUGE metric by examples. Medium.
https://medium.com/nlplanet/two-minutes-nlp-learn-the-rouge-metric-by-examples-f179cc285499

BERTScore. (n.d.). Machine Translate. https://machinetranslate.org/bertscore

Rae, & Rae. (n.d.). inteligencia | Diccionario del estudiante. «Diccionario Del Estudiante».
https://www.rae.es/diccionario-estudiante/inteligencia

Farías, G. (2024, January 16). Coherencia - Concepto, factores, cohesión y adecuación. Concepto.
https://concepto.de/coherencia/

Rockets, R. (2019, November 7). La importancia de la fluidez. Colorín Colorado.
https://www.colorincolorado.org/es/articulo/la-importancia-de-la-fluidez

Gil, J. M. (2018). Qué es la creatividad lingüística: una explicación neurocognitiva a partir de nombres de comercios de Mar del Plata. Logos, 28(1), 116–134. https://doi.org/10.15443/rl2810

BERT Score - a Hugging Face Space by evaluate-metric. (n.d.).
https://huggingface.co/spaces/evaluate-metric/bertscore

BLEU - a Hugging Face Space by evaluate-metric. (n.d.). https://huggingface.co/spaces/evaluate-metric/bleu

ROUGE - a Hugging Face Space by evaluate-metric. (n.d.). https://huggingface.co/spaces/evaluate-metric/rouge

WikiSplit - a Hugging Face Space by evaluate-metric. (n.d.).
https://huggingface.co/spaces/evaluate-metric/wiki_split

SARI - a Hugging Face Space by evaluate-metric. (n.d.). https://huggingface.co/spaces/evaluate-metric/sari

Exact Match - a Hugging Face Space by evaluate-metric. (n.d.). https://huggingface.co/spaces/evaluate-metric/exact_match

SacreBLEU - a Hugging Face Space by evaluate-metric. (n.d.). https://huggingface.co/spaces/evaluate-metric/sacrebleu

google/flan-t5-base · Hugging Face. (2023, June 11). https://huggingface.co/google/flan-t5-base

mistralai/Mistral-7B-Instruct-v0.2 · Hugging Face. (n.d.). https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2

OpenAI Platform. (n.d.). https://platform.openai.com/docs/overview

Modelos de lenguaje extenso (LLM) con la IA de Google. (n.d.). Google Cloud. https://cloud.google.com/ai/llms?hl=es

Introduction | 🔗 LangChain. (n.d.). https://python.langchain.com/v0.2/docs/introduction/

What is a Transformer Model? | IBM. (n.d.). https://www.ibm.com/topics/transformer-model

Merritt, R. (2022, September 16). What is a transformer model? | NVIDIA Blogs. NVIDIA Blog. https://blogs.nvidia.com/blog/what-is-a-transformer-model/

Instituto de Ingeniería del Conocimiento. (2024, February 14). Procesamiento del Lenguaje Natural. Instituto De Ingeniería Del Conocimiento. https://www.iic.uam.es/inteligencia-artificial/procesamiento-del-lenguaje-natural/

Procesamiento de lenguaje natural (PLN). (n.d.). www.cognizant.com. https://www.cognizant.com/es/es/glossary/natural-language-processing

¿Qué es el procesamiento de lenguaje natural? - Explicación del procesamiento de lenguaje natural - AWS. (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/nlp/

Grandes Modelos de Lenguaje (Large Language Models) | Codificando Bits. (n.d.). Codificando Bits. https://www.codificandobits.com/blog/grandes-modelos-de-lenguaje/

Equipo editorial, Etecé. (2021, August 5). Semántica - Concepto, componentes y ejemplos. Concepto. https://concepto.de/semantica/

SLU, Z. C. (n.d.). *Relevancia | Zorraquino*. Zorraquino. https://www.zorraquino.com/diccionario/marketing-digital/que-es-relevancia.html

Jimenez, A. (2019, February 28). *FLUIDEZ EN ESPAÑOL | Aprende hablando*. Aprende Hablando. https://aprendehablando.com/la-fluidez-en-espanol/

Gil, J. M. (2018). Qué es la creatividad lingüística: una explicación neurocognitiva a partir de nombres de comercios de Mar del Plata. *Logos*, *28*(1), 116–134. https://doi.org/10.15443/rl2810

Foqum Analytics, Empowers your success. (2023, November 17). *N-Grama | FOQUM*. FOQUM. https://foqum.io/blog/termino/n-grama/

SPSS Modeler Subscription. (2021, August 17).

https://www.ibm.com/docs/es/spss-modeler/saas?topic=networks-neural-model