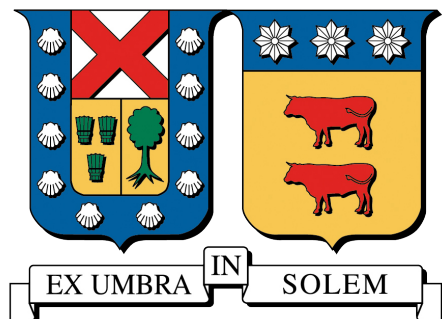


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAISO - CHILE



"PREPARARIA: RECETAS PERSONALIZADAS GENERADAS POR INTELIGENCIA ARTIFICIAL."

BRITNEY SANDON GUERRERO

**MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE
INGENIERA CIVIL EN ELECTRÓNICA**

PROFESOR GUÍA : DR. WERNER CREIXELL
PROFESOR CORREFERENTE : DR. PATRICIO OLIVARES

AGOSTO 2024

A mi papá . . .

AGRADECIMIENTOS

Aún recuerdo cuando dejé mi hogar y mi familia en el norte para comenzar este desafío. Sabía que iba a ser doloroso extrañarlos tanto. Sin embargo, la vida tenía preparado cruzarme con un montón de personas maravillosas, a quienes hoy puedo llamar amigos y familia.

Quiero agradecer a mis amigos, quienes fueron mi fuente inagotable de apoyo, cariño y contención. Gracias por tenderme una mano cuando lo necesité, y espero haber estado a la altura de una amistad tan bella como la que me entregaron.

En especial, agradecer a Matías y Diego, mis otras dos neuronas; gracias por acompañarme en este recorrido lleno de risas, llanto, desveladas, conversaciones profundas y tecitos con huevo y palta. Gracias por crecer conmigo.

A mi familia, que a pesar de la distancia, la adversidad y las pérdidas, hemos logrado salir adelante.

A mi papi, quien sin tener nada, me educó y amó por sobre todas las cosas. Espero algún día devolverte todo lo bueno que me has dado.

Último, pero no menos importante, Ignacio. Gracias por amarme en todas mis facetas, gracias por ser la familia que elegí y mi mejor compañía junto a nuestra Mantequilla.



PrepararIA: Recetas personalizadas generadas por Inteligencia Artificial.

Britney Vaitiare Sandon Guerrero

Memoria para optar al título de Ingeniera Civil en Electrónica, mención Computadores,
submención Telecomunicaciones.

Universidad Técnica Federico Santa María

Profesor Guía: Dr. Werner Creixell Fuentes
Profesor Correferente: Dr. Patricio Olivares Roncagliolo

AGOSTO 2024

Resumen

El presente trabajo se centra en el desarrollo de un sistema de recomendación de recetas basado en el modelo de lenguaje preentrenado Mistral 7B, ajustado con un conjunto de datos de recetas. El sistema contempla dos escenarios principales: el primero sugiere recetas en función de los ingredientes disponibles, mientras que el segundo proporciona recetas según las preferencias alimentarias, el tipo de platillo y el nivel de dificultad deseado para la preparación.

Para optimizar los recursos disponibles, se emplearon técnicas de ajuste eficiente de parámetros, particularmente QLoRA (Quantized Low-Rank Adaptation). Esta técnica permite adaptar modelos de gran escala como el Mistral 7B a tareas específicas sin necesidad de reentrenar completamente el modelo, reduciendo el costo computacional asociado.

El sistema resultante, ofrece una interfaz sencilla para que los usuarios interactúen con el modelo en ambos escenarios propuestos. Su evaluación incluye métricas cuantitativas como ROUGE y análisis cualitativos basados en la percepción de los usuarios.



PrepararIA: Personalized Recipes Generated by Artificial Intelligence.

Britney Vaitiare Sandon Guerrero

Thesis for the fulfillment of the B.S. in Electronic Engineering, major in Computer Electronics, minor in Telecommunications (6 year program).

Universidad Técnica Federico Santa María

Advisor: Dr. Werner Creixell Fuentes
Co-Advisor: Dr. Patricio Olivares Roncagliolo

AUGUST 2024

Abstract

The present work focuses on the development of a recipe recommendation system based on the pre-trained Mistral 7B language model, fine-tuned with a recipe dataset. The system addresses two main scenarios: the first suggests recipes based on available ingredients, while the second provides recipes according to dietary preferences, dish type, and the desired difficulty level.

To optimize the available resources, parameter-efficient fine-tuning techniques were employed, particularly QLoRA (Quantized Low-Rank Adaptation). This technique allows large-scale models like Mistral 7B to be adapted to specific tasks without the need for full retraining, thereby reducing the associated computational cost.

The resulting system offers a simple interface for users to interact with the model in both proposed scenarios. Its evaluation includes quantitative metrics such as ROUGE and qualitative analysis based on user perception.

Glosario

FINE-TUNING Proceso de ajuste de un modelo preentrenado para una tarea específica utilizando un conjunto de datos particular.

GPT Modelo de lenguaje basado en la arquitectura Transformer que es preentrenado en grandes cantidades de datos textuales para generar texto coherente y relevante.

INFERENCIA Proceso mediante el cual un modelo de aprendizaje automático o de lenguaje genera predicciones o resultados a partir de nuevos datos de entrada, basándose en el conocimiento adquirido durante su entrenamiento.

LATENCIA Tiempo de retraso entre la entrada de un comando o solicitud y la respuesta efectiva del sistema.

LLMs Modelos de lenguaje entrenados en grandes cantidades de datos textuales, diseñados para comprender, procesar y generar lenguaje natural con alta precisión.

N-GRAMA Secuencia continua de n elementos (como palabras o caracteres) extraídos de un texto. En el procesamiento del lenguaje natural, los n -gramas se utilizan para modelar la probabilidad de aparición de secuencias de palabras o caracteres en un texto.

NER Técnica de procesamiento del lenguaje natural que identifica y clasifica entidades importantes en un texto, como nombres de personas, lugares, organizaciones o fechas.

PRECISION Métrica que indica la proporción de predicciones correctas que realmente son relevantes.

Índice general

1	Introducción	9
2	Estado del Arte	10
2.1	Modelos de Lenguaje Preentrenados para la Generación de Texto	10
2.2	RecipeGPT	11
2.3	EvoRecipes	11
2.4	Smart Cuisine	12
3	Desarrollo de la Solución	13
3.1	Descripción del Sistema	13
3.2	Técnicas para el Ajuste Eficiente de Parámetros en LLMs	14
3.2.1	LoRA: Low-Rank Adaptation	15
3.2.2	QLoRA: Quantized Low-Rank Adaptation	16
3.3	Modelo	16
3.3.1	Fine-tuning del Modelo	17
3.4	Dataset	18
3.5	Interfaz	21
4	Resultados	23
4.1	Operación del Sistema	23
4.2	Evaluación del Sistema	24
4.2.1	Rouge: Recall-Oriented Understudy for Gisting Evaluation	24
4.2.2	Percepción del usuario	27
5	Conclusión	32

Índice de cuadros

3.1	Comparación de modelos y su rendimiento en diferentes tareas. Adaptado de [1].	17
3.2	Hiperparámetros utilizados en el entrenamiento de los modelos.	18
3.3	Descripción de la estructura del conjunto de datos RecipeNLG.	18
4.1	Evaluación del modelo ajustado para el escenario 1 mediante las métricas ROUGE.	25
4.2	Evaluación del modelo ajustado para el escenario 2 mediante las métricas ROUGE.	26

Índice de figuras

2.1	Ilustración basada en la operación de RecipeGPT. Adaptado de [2].	11
3.1	Diagrama de flujo del escenario 1 en PrepararIA.	13
3.2	Diagrama de flujo del escenario 2 en PrepararIA.	14
3.3	Reparametrización e inyección de matrices implementada por LoRA. Adaptado de [3].	15
3.4	Formato de instrucción utilizado para el fine-tuning del modelo Mistral-7B-Instruct-v0.3.	17
3.5	Registro ejemplo del conjunto de datos RecipeNLG.	19
3.6	Ejemplo de la estructura del conjunto de datos adaptado para el escenario 1. . .	19
3.7	Ejemplo de la estructura del conjunto de datos adaptado para el escenario 2. . .	20
3.8	Conjuntos de datos adaptados para los escenarios definidos, alojados en Hugging Face.	20
3.9	Modelos ajustados para generar recetas según los escenarios definidos, alojados en Hugging Face.	21
3.10	Página de inicio de la aplicación web PrepararIA.	21
3.11	Página web del escenario 1: Recetas basadas en ingredientes disponibles.	22
3.12	Página web del escenario 2: Recetas basadas en preferencias y restricciones alimentarias.	22
4.1	Vista de la operación del sistema en el escenario 1.	23
4.2	Vista de la operación del sistema en el escenario 2.	24
4.3	Código utilizado para el cálculo de las métricas ROUGE.	25
4.4	Ejemplo del texto generado y texto de referencia usados para evaluar el modelo del escenario 1.	25
4.5	Ejemplo del texto generado y texto de referencia usados para evaluar el modelo del escenario 2.	26
4.6	Gráfico de la percepción del usuario sobre la navegación en la aplicación.	27
4.7	Gráfico de la percepción del usuario sobre la receta generada en el escenario 1. .	28
4.8	Gráfico de la percepción del usuario sobre la receta generada en el escenario 2. .	28
4.9	Gráfico de la percepción del usuario sobre la claridad y fluidez de las recetas generadas.	29
4.10	Gráfico de la percepción del usuario sobre la creatividad de las recetas generadas. .	29
4.11	Gráfico de la percepción del usuario sobre la probabilidad de preparar las recetas generadas.	30
4.12	Gráfico de la percepción del usuario sobre el tiempo de respuesta del sistema. . .	30
4.13	Gráfico que indica la disposición de los usuarios a usar la aplicación.	31

1 | Introducción

La alimentación no solo es una necesidad básica para la supervivencia, sino que también es fundamental para el bienestar general y la calidad de vida de las personas. En la actualidad, el desperdicio de alimentos ha alcanzado niveles alarmantes, con aproximadamente un tercio de la producción global terminando en desperdicio [4]. Este problema no solo representa una pérdida económica significativa, sino que también tiene graves consecuencias ambientales [5]. Ante la creciente conciencia sobre las diversas necesidades alimentarias, surge la necesidad de soluciones que faciliten la planificación de comidas y ayuden a minimizar el desperdicio desde el momento en que se adquieren los alimentos.

En este contexto, “PrepararIA” se presenta como una solución que busca optimizar los recursos alimenticios disponibles en casa, además de ofrecer recetas considerando las restricciones dietéticas del usuario. Utilizando inteligencia artificial, la plataforma ofrece recomendaciones de recetas personalizadas que responden a la pregunta común: “¿Qué puedo cocinar?”, buscando orientar al usuario a la compra eficiente de alimentos y uso óptimo de ingredientes disponibles. Las recetas son un ejemplo claro de lenguaje natural estructurado, ya que siguen un formato predefinido con listas de ingredientes y pasos secuenciales. Esta estructura facilita su procesamiento por modelos de lenguaje de gran tamaño (LLMs) [6], como los utilizados en “PrepararIA”. Al trabajar con un formato regular, los LLMs pueden identificar patrones consistentes, lo que les permite generar recomendaciones precisas y coherentes.

Los modelos disponibles son de propósito general y requieren ser ajustados para adaptarse a la tarea específica de recomendar recetas. Para esto, se ha adaptado un conjunto de datos existente que incluye diferentes tipos de recetas para entrenar y ajustar el modelo. Este proceso consume una gran cantidad de recursos, por lo que se han empleado técnicas como la adaptación de bajo rango y la cuantización [7], reduciendo los recursos computacionales necesarios y asegurando que el sistema sea accesible y capaz de ofrecer recomendaciones adaptadas a las necesidades específicas de cada usuario, sin requerir una inversión significativa de recursos adicionales.

Con el objetivo de mejorar la interacción con los usuarios, se ha desarrollado una interfaz intuitiva que facilita el uso del sistema.

Finalmente, la plataforma se evalúa utilizando métricas cuantitativas, como ROUGE, y medidas cualitativas, como la satisfacción y percepción del usuario.

2 | Estado del Arte

2.1 Modelos de Lenguaje Preentrenados para la Generación de Texto

Los modelos de lenguaje preentrenados (PLMs) han revolucionado la generación de texto, permitiendo crear contenido coherente y contextualizado a partir de grandes volúmenes de datos no etiquetados [8]. Estos modelos, como GPT y BERT [9], han demostrado su eficacia en tareas como la traducción automática, el resumen de textos y la generación de diálogos.

El preentrenamiento de los PLMs les proporciona un conocimiento profundo de las relaciones semánticas complejas del lenguaje, facilitando su ajuste a nuevas tareas con menos datos. Esto optimiza su capacidad para generar texto de alta calidad y mejora su aplicabilidad en diversos dominios.

En cuanto a la evaluación del texto generado, se identifican limitaciones en las métricas automáticas como BLEU [10] y ROUGE [11], las cuales se basan principalmente en la coincidencia de n-gramas entre el texto generado y el de referencia. Aunque son útiles para tareas como la traducción automática, estas métricas no capturan adecuadamente aspectos más complejos, como la coherencia del flujo narrativo, la precisión gramatical o la continuidad temática. Debido a estas limitaciones, el uso de evaluaciones humanas sigue siendo fundamental para medir aspectos como la coherencia, creatividad y utilidad práctica del texto generado.

En cuanto a la escalabilidad, se destacan tanto las ventajas como los desafíos de los modelos de gran tamaño, como GPT-3 [12], que contiene miles de millones de parámetros. Si bien el incremento en la cantidad de parámetros ha demostrado mejorar significativamente la calidad del texto generado, también conlleva un aumento considerable en los costos computacionales. Este factor no solo afecta los recursos necesarios para el entrenamiento y la inferencia, sino que también plantea desafíos en términos de sostenibilidad y accesibilidad, especialmente para aplicaciones que requieren personalización a gran escala o implementación en tiempo real.

En resumen, encontrar el equilibrio entre la generación de texto de alta calidad y la viabilidad práctica de implementar estos modelos en entornos con recursos limitados sigue siendo un desafío importante.

2.2 RecipeGPT

RecipeGPT [2] es un sistema de generación y evaluación de recetas de cocina basado en el ajuste de un modelo de propósito general. Utiliza el modelo pre-entrenado GPT-2 [13], un LLM con 1500 millones de parámetros, que ha sido ajustado con un dataset amplio y variado que contiene más de un millón de recetas [14].

RecipeGPT ofrece dos modos de generación de texto: la creación de instrucciones de cocina a partir de un título de receta con una lista de ingredientes, y la generación de ingredientes a partir del título y las instrucciones de cocina. Además, el sistema permite a los usuarios evaluar la calidad de las recetas generadas y almacenar aquellas que el usuario considere relevantes para acceder a ellas en cualquier momento.

Este sistema ha demostrado ser efectivo en la generación de textos de recetas de cocina, resaltando cómo modelos generativos de propósito general pueden abordar tareas específicas con un alto rendimiento luego de un proceso de fine-tuning. Los resultados presentados por RecipeGPT demuestran el potencial y la viabilidad de proyectos similares, como PrepararIA.

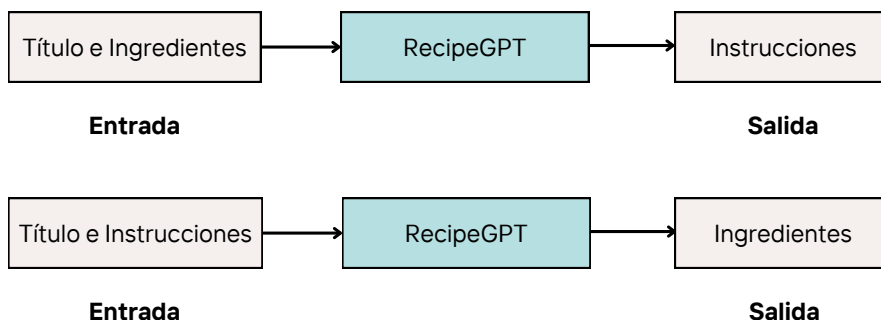


Figura 2.1: Ilustración basada en la operación de RecipeGPT. Adaptado de [2].

2.3 EvoRecipes

EvoRecipes [15] es un framework que combina algoritmos evolutivos y LLMs para crear recetas personalizadas, considerando las preferencias del usuario. A diferencia de los métodos tradicionales que se basan únicamente en patrones estadísticos, EvoRecipes integra un enfoque semántico mediante el uso de ontologías como RecipeOn y grafos de conocimiento como RecipeKG.

El empleo de ontologías y grafos de conocimiento en este framework permite estructurar la información culinaria de manera que las relaciones entre ingredientes, acciones y procedimientos se representen de forma coherente. RecipeOn define las clases, propiedades y relaciones entre los diferentes componentes de una receta, mientras que RecipeKG almacena un vasto conjunto de recetas codificadas en formato RDF (Resource Description Framework), capturando no solo los ingredientes y las instrucciones, sino también las interacciones entre ellos y su contexto en la preparación de un platillo.

El framework emplea un algoritmo genético inspirado en la selección natural para explorar un amplio espacio de soluciones codificadas en RDF, aplicando operadores de mutación y cruce para generar nuevas combinaciones de ingredientes y procedimientos. Este proceso evolutivo se guía por una función de aptitud multiobjetivo que considera factores clave como la novedad, simplicidad, atractivo visual y viabilidad de las recetas generadas, asegurando que estas se alineen con las preferencias del usuario. Finalmente, las recetas en formato RDF son convertidas a texto natural mediante modelos de lenguaje, lo que facilita su comprensión y uso por parte de los usuarios.

EvoRecipes integra de manera efectiva los LLMs dentro de un proceso más amplio que incluye algoritmos evolutivos y un enfoque semántico robusto, permitiendo la generación de recetas personalizadas que son coherentes y adaptadas a las necesidades y gustos individuales.

2.4 Smart Cuisine

Smart Cuisine [16] es un sistema que utiliza inteligencia artificial para apoyar la planificación de comidas y promover prácticas más sostenibles.

El sistema incluye un generador de recetas impulsado por un modelo de Facebook Meta AI, que sugiere nuevas combinaciones a partir de los ingredientes disponibles, permitiendo a los usuarios aprovechar mejor lo que ya tienen en casa. Además, cuenta con una calculadora de proporciones que ajusta la cantidad y el costo de los ingredientes según las necesidades del usuario, orientando la planificación de comidas de forma que las recetas generadas se adapten al poder adquisitivo del usuario.

Un aspecto clave de Smart Cuisine es su servicio de chat impulsado por OpenAI, que responde a preguntas sobre nutrición y salud en lenguaje natural. Este servicio busca facilitar que los usuarios tomen decisiones más informadas sobre su alimentación, promoviendo hábitos más saludables.

Smart Cuisine se distingue por su enfoque integral, que combina la generación de recetas con la sostenibilidad y la salud, lo que lo hace particularmente relevante para proyectos como PrepararIA, que buscan personalizar la experiencia culinaria mientras fomentan el cuidado del medio ambiente.

3 | Desarrollo de la Solución

En este capítulo se describe la metodología empleada para el desarrollo de la solución propuesta. Se abordan las distintas etapas del proyecto, desde la descripción del sistema hasta el despliegue final de la aplicación.

3.1 Descripción del Sistema

A continuación, se describen los dos escenarios principales que han guiado el diseño e implementación de la solución. Estos escenarios explican cómo los usuarios interactúan con el sistema para generar recetas.

Escenario 1: Recetas Basadas en Ingredientes Disponibles.

En el primer escenario, el usuario se comunica con PrepararIA para recibir una recomendación de receta basada en los ingredientes disponibles. El proceso comienza con la introducción de estos ingredientes a través de la interfaz del sistema, donde deben ser ingresados en un formato de Reconocimiento de Entidades Nombradas (NER). Esto significa que los ingredientes se registran únicamente por su nombre, sin incluir cantidades ni unidades de medida. Utilizando un modelo ajustado para esta tarea, la aplicación procesa la entrada y genera una receta que se puede preparar con los ingredientes ingresados. La receta incluye el título, los ingredientes con sus medidas respectivas y las instrucciones de cocina.

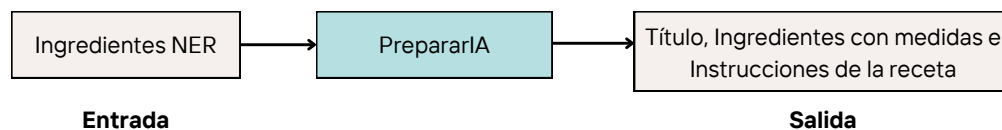


Figura 3.1: Diagrama de flujo del escenario 1 en PrepararIA.

Escenario 2: Recetas Basadas en Preferencias y Restricciones Alimentarias.

En el segundo escenario, “PrepararIA” ofrece al usuario una recomendación de receta basada en sus preferencias alimentarias o restricciones dietéticas, tipo de plato y nivel de habilidad en la cocina. El proceso comienza cuando el usuario ingresa esta información a través de la interfaz. Utilizando un modelo ajustado para esta tarea, el sistema procesa la entrada y genera una receta acorde a las especificaciones del usuario incluyendo título, ingredientes e instrucciones de cocina.

Dentro de los tipos de dieta se han considerado:

- Omnívoro, Vegetariano, Vegano, Celíaco y Diabético.

Los tipos de preparaciones incluidos son:

- Desayuno, Almuerzo, Cena y Postre.

Finalmente, se han considerado tres niveles de habilidad en la cocina:

- Principiante, Intermedio y Avanzado.

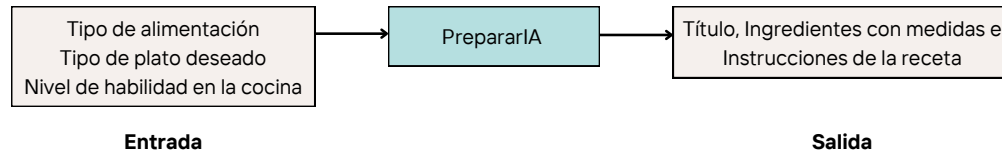


Figura 3.2: Diagrama de flujo del escenario 2 en PrepararIA.

3.2 Técnicas para el Ajuste Eficiente de Parámetros en LLMs

El proceso de fine-tuning de los LLMs es extremadamente costoso desde el punto de vista computacional y poco práctico debido a la cantidad masiva de parámetros que conforman estos modelos, lo que hace inviable ajustar directamente sus billones de parámetros. En este contexto, las técnicas de Parameter-Efficient Fine-Tuning (PEFT) [17] juegan un papel fundamental para realizar este proceso en entornos con recursos limitados.

PEFT se enfoca en reducir los elevados requerimientos computacionales al entrenar solo una pequeña fracción de los pesos del modelo. Estos pueden ser un subconjunto de los parámetros existentes o un conjunto de nuevos parámetros añadidos al modelo. Los métodos propuestos varían en cuanto a la cantidad de pesos que necesitan ser ajustados, así como en el uso de memoria, la velocidad de entrenamiento, la calidad del modelo resultante y posibles incrementos en los tiempos de inferencia.

Entre los métodos PEFT se encuentran:

- **Additive:** Este método introduce nuevos parámetros en las capas o subcapas del modelo. Durante el entrenamiento, solo se ajustan estos nuevos elementos, mientras que los parámetros originales permanecen sin cambios.
- **Adapters:** Consiste en introducir pequeñas redes totalmente conectadas después de las subcapas en los módulos de atención y de propagación directa (feed-forward) de la red Transformer.
- **Soft Prompts:** En lugar de ajustar los parámetros del modelo principal, se ajustan vectores de entrada adicionales (prompts) que guían el comportamiento del modelo sin cambiar su estructura interna.

- **Selective:** Este método selecciona y ajusta solo un subconjunto de los parámetros del modelo, identificados como los más relevantes para la tarea específica utilizando diversos criterios.
- **Reparametrization-based:** Este método reparametriza los pesos del modelo al descomponerlos en productos de matrices más pequeñas, que son las únicas ajustadas durante el proceso de fine-tuning.

3.2.1 LoRA: Low-Rank Adaptation

Dentro de la categoría *reparametrization-based*, destaca LoRA [3]. Este enfoque congela los pesos preentrenados del modelo e inyecta matrices de bajo rango en las capas proyectivas, como los módulos de atención y feedforward. Durante el fine-tuning, en lugar de ajustar todos los parámetros del modelo, LoRA modifica únicamente las matrices inyectadas, en las cuales recae la actualización de los pesos necesaria para que el modelo se adapte a la nueva tarea. Aunque este método se ha probado principalmente en las capas de atención y feedforward, es aplicable a otras capas del modelo.

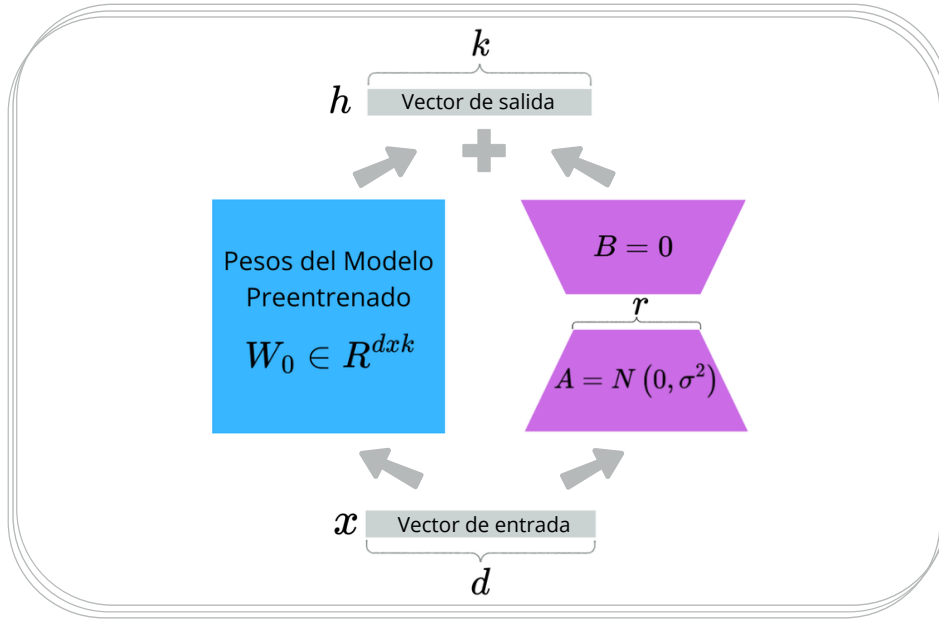


Figura 3.3: Reparametrización e inyección de matrices implementada por LoRA. Adaptado de [3].

La siguiente ecuación muestra cómo se integran los pesos del modelo original con las matrices inyectadas por LoRA en las capas seleccionadas durante el entrenamiento:

$$W'_0 = W_0 + \Delta W = W_0 + BA \quad (3.1)$$

donde W_0 son los pesos preentrenados, ΔW es la actualización modelada como el producto de B (de dimensiones $\mathbb{R}^{d \times r}$) y A (de dimensiones $\mathbb{R}^{r \times k}$), siendo r el rango seleccionado. Aquí, d representa la dimensión de la entrada al modelo (o capa) y k es la dimensión de salida de los pesos originales W_0 . La matriz A se inicializa con una distribución normal para permitir que sus

valores varíen aleatoriamente alrededor de cero, facilitando el aprendizaje de patrones desde el comienzo del entrenamiento. En cambio, la matriz B se inicializa en cero para garantizar que la contribución de ΔW comience nula, y solo con el proceso de optimización la matriz B se ajuste, lo que permite una actualización controlada de los pesos durante el fine-tuning. Finalmente, W'_0 representa los pesos resultantes del modelo adaptado a la tarea específica.

LoRA no introduce latencia adicional en la inferencia, ya que integra las matrices de actualización directamente en los pesos preentrenados, sin pasos computacionales extra ni cambios en la estructura del modelo.

Un análisis comparativo de varios métodos PEFT, presentado en [3], demostró que LoRA ofrece un rendimiento comparable al de los modelos con ajuste total de parámetros, pero utilizando solo una fracción de los pesos.

Finalmente, es importante destacar que LoRA permite cambiar rápidamente entre diferentes tareas al intercambiar las matrices de bajo rango específicas, sin alterar la estructura base del modelo.

3.2.2 QLoRA: Quantized Low-Rank Adaptation

A pesar de que LoRA reduce de manera significativa los recursos necesarios para el fine-tuning de LLMs, todavía puede resultar insuficiente en plataformas con recursos limitados, como Google Colab, donde tanto la memoria como la capacidad de procesamiento son reducidas.

Por esta razón, se ha investigado la integración de LoRA con técnicas de cuantización, como la presentada en QLoRA [7]. La cuantización es una técnica utilizada para reducir el tamaño de los modelos al representar los pesos con menos bits.

En el contexto de QLoRA, se utiliza una cuantización de 4 bits conocida como NormalFloat (NF4), que reduce los pesos de precisión completa (32 bits) a una forma más compacta de 4 bits, disminuyendo drásticamente la memoria requerida para almacenar el modelo. NF4 es una mejora sobre los formatos de cuantización previos, optimizada teóricamente para datos distribuidos normalmente, como los pesos preentrenados de redes neuronales.

3.3 Modelo

Para seleccionar un modelo adecuado para fine-tuning en entornos con recursos limitados, se determinó que era esencial optar por uno con una baja cantidad de parámetros y buen rendimiento. Mistral AI ha comparado su modelo **Mistral 7B** [1] con otros modelos de lenguaje de tamaño similar, como LLaMA 2 7B y LLaMA 2 13B, en una amplia gama de tareas (ver cuadro 3.1). Estas tareas abarcan la comprensión de textos en diferentes áreas, la generación de respuestas coherentes en diversos escenarios, y la capacidad de responder con precisión a preguntas complejas, entre otras.

Debido a su destacado desempeño y su capacidad para ajustarse a tareas específicas utilizando técnicas PEFT, se eligió Mistral 7B para el desarrollo de la solución presentada en este documento.

Modelo	Nº Parámetros	Licencia	MMLU	HellaSwag	NQ
Mistral 7B	7.300 Mill.	Apache 2.0 License	60.1 %	81.3 %	28.8 %
LLaMA 2 7B	7.000 Mill.	LLaMA 2 Community License	44.4 %	77.1 %	24.7 %
LLaMA 2 13B	13.000 Mill.	LLaMA 2 Community License	55.6 %	80.7 %	29.0 %
Code-Llama 7B	7.000 Mill.	LLaMA 2 Community License	36.9 %	62.9 %	11.0 %

Cuadro 3.1: Comparación de modelos y su rendimiento en diferentes tareas. Adaptado de [1].

Mistral 7B está basado en la arquitectura Transformer [18], y ha sido previamente optimizado mediante una serie de técnicas en sus capas de atención, que incluyen:

- **Grouped-Query Attention (GQA):** Esta técnica agrupa las consultas, permitiendo que varias palabras compartan las mismas claves y valores para determinar su relevancia en una secuencia. Esto facilita que el modelo evalúe la importancia de estas palabras colectivamente, en lugar de hacerlo de forma individual. Al reducir el número de cálculos necesarios en el mecanismo de atención, GQA [19] disminuye el costo de inferencia, optimizando así el uso de memoria y tiempo de procesamiento.
- **Sliding-Window Attention (SWA):** Esta técnica [20] optimiza el procesamiento de secuencias largas en modelos de lenguaje al dividir el texto en ventanas de tamaño fijo. Cada palabra en una ventana solo interactúa con palabras cercanas. A medida que el modelo avanza, la ventana se desplaza, lo que permite una atención eficiente y evita el uso excesivo de memoria.

3.3.1 Fine-tuning del Modelo

Para este proceso, se eligió la versión Mistral-7B-Instruct-v0.3, versión que ha sido entrenada para seguir instrucciones y proporcionar respuestas precisas en contextos especificados por el usuario. Este modelo fue preentrenado con información en inglés y utiliza un formato estructurado que incluye secciones claras para la instrucción, la entrada y la respuesta generada, delimitadas por los tokens <s> y </s>. Durante el fine-tuning, presentar ejemplos con un formato estructurado a los modelos generativos es fundamental para que aprendan a generar respuestas adecuadas en función de entradas bien definidas, asegurando así precisión y coherencia en sus respuestas.

La estructura definida en este desarrollo es la siguiente:

```
"<s> Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

### Instruction:
{}
### Input:
{}
### Output:
{} </s>"
```

Figura 3.4: Formato de instrucción utilizado para el fine-tuning del modelo Mistral-7B-Instruct-v0.3.

Para aplicar técnicas como QLoRA en el modelo Mistral 7B, se utiliza la herramienta Unsloth, accesible a través de la plataforma de Hugging Face [21]. Unsloth es una plataforma especializada en la optimización del entrenamiento y fine-tuning de LLMs, lo que permite mejorar la eficiencia del modelo y reducir significativamente el uso de recursos durante el proceso de entrenamiento.

El ajuste del modelo se realizó en Google Colaboratory utilizando la GPU T4 gratuita que proporciona la plataforma. Para este proceso, se han definido los siguientes hiperparámetros para el entrenamiento de los modelos de ambos escenarios propuestos:

Hiperparámetro	Valor
Epochs	3
Learning Rate	2e-5
Batch Size	8
LoRA Rank	16
LoRA α	16

Cuadro 3.2: Hiperparámetros utilizados en el entrenamiento de los modelos.

3.4 Dataset

Al ajustar modelos generativos para tareas específicas, la calidad y coherencia del conjunto de datos son factores clave para la efectividad de las respuestas generadas.

Por ello, se utilizó RecipeNLG [22], una colección de más de dos millones de recetas disponible en Hugging Face. Cada receta está organizada en campos que detallan ingredientes, instrucciones y otros datos relevantes, lo que facilita su adaptación a diversos escenarios utilizando la librería Pandas [23].

Campo	Descripción
id	Identificador único de cada receta.
title	Título de la receta.
ingredients	Listado de ingredientes y sus respectivas cantidades necesarias para la elaboración de la receta.
directions	Instrucciones detalladas para la preparación de la receta.
link	Enlace a la fuente original de la receta.
source	Indica el origen de los datos, ya sea “Gathered” (recopiladas de la web) o “Recipes1M” (extraídas del dataset Recipe1M+).
ner	Lista de Ingredientes en formato NER.

Cuadro 3.3: Descripción de la estructura del conjunto de datos RecipeNLG.

Ejemplo del conjunto del datos:

```
{'id': 0,
 'title': 'No-Bake Nut Cookies',
 'ingredients': ['1 c. firmly packed brown sugar', '1/2 c. evaporated milk', '1/2
                tsp. vanilla', '1/2 c. broken nuts (pecans)', '2 Tbsp. butter or
                margarine', '3 1/2 c. bite size shredded rice biscuits'],
 'directions': ['In a heavy 2-quart saucepan, mix brown sugar, nuts, evaporated
                milk and butter or margarine.', 'Stir over medium heat until mixture
                bubbles all over top.', 'Boil and stir 5 minutes more. Take off
                heat.', 'Stir in vanilla and cereal; mix well.', 'Using 2 teaspoons,
                drop and shape into 30 clusters on wax paper.', 'Let stand until firm,
                about 30 minutes.'],
 'link': 'www.cookbooks.com/Recipe-Details.aspx?id=44874',
 'source': 0,
 'ner': ['brown sugar', 'milk', 'vanilla', 'nuts', 'butter',
         'bite size shredded rice biscuits']}
```

Figura 3.5: Registro ejemplo del conjunto de datos RecipeNLG.

Conjunto de datos para adaptar el modelo del escenario 1

Para adaptar RecipeNLG a las necesidades del primer escenario, se realizaron los siguientes pasos:

1. **Selección de muestra:** Se tomó una muestra aleatoria de 5000 recetas, incluyendo sólo los campos NER, title, ingredients y directions.
2. **Construcción del DataFrame:** Se creó un nuevo DataFrame *df* con tres campos:
 - input: Contiene la lista de ingredientes en formato NER.
 - output: Contiene una lista generada a partir de la combinación de los campos title, ingredients y directions mediante una función lambda. Ya que se busca que el modelo genere una respuesta con ese formato.
 - instruction: Contiene una instrucción estándar que especifica la tarea a realizar, en este caso: “Create a recipe using these ingredients”.

	input	output	instruction
0	['Marshmallow Creme', 'cream cheese']	[Apple Dip, ['1 (7 oz.) jar Marshmallow Creme'...	Create a recipe using this ingredients
1	['russet', 'low-fat milk', 'butter', 'wasabi p...	[Wasabi Whipped Potatoes, ['1 1/2 lb. russet o...	Create a recipe using this ingredients
2	['bread', 'brown sugar', 'regular vanilla pudd...	[Caramel Breakfast Rolls, ['2 (1 lb.) loaves f...	Create a recipe using this ingredients
3	['cream', 'eggs', 'pepper', 'salt', 'pork saus...	[Egg And Sausage Muffins , ['1/4 cup cream (ha...	Create a recipe using this ingredients
4	['lean ground beef', 'onion', 'pintos', 'kidne...	[Taco Soup, ['2 lb. lean ground beef', '1 smal...	Create a recipe using this ingredients

Figura 3.6: Ejemplo de la estructura del conjunto de datos adaptado para el escenario 1.

Conjunto de datos para adaptar el modelo del escenario 2

Para adaptar RecipeNLG a las necesidades del segundo escenario, se realizaron los siguientes pasos:

1. **Selección de muestra:** Se tomó una muestra aleatoria de 5000, incluyendo sólo los campos `title`, `ingredients` y `directions`.
2. **Clasificación de Recetas:** Se utilizó la API de OpenAI, específicamente el modelo `gpt-3.5-turbo` [24], para clasificar las recetas según los criterios de *Diet Type*, *Meal Type* y *Skill Level*.
3. **Construcción del DataFrame:** Se creó un nuevo DataFrame `df` con tres campos:
 - `input`: Esta columna contiene la clasificación de la receta en el formato “Diet Type: <type>, Meal Type: <type>, Skill Level: <level>”.
 - `output`: Contiene una lista generada a partir de la combinación de los campos `title`, `ingredients` y `directions` mediante una función `lambda`. Ya que se busca que el modelo genere una respuesta con ese formato.
 - `instruction`: Contiene una instrucción estándar que especifica la tarea a realizar, en este caso: “Create a recipe based on the following”.

	input	output	instruction
0	Diet Type: Vegetarian, Meal Type: Dessert, Ski...	['Apple Dip', ['1 (7 oz.) jar Marshmallow Cre...	Create a recipe based on the following
1	Diet Type: Omnivore, Meal Type: Dinner, Skill ...	['Wasabi Whipped Potatoes', ['1 1/2 lb. russe...	Create a recipe based on the following
2	Diet Type: Omnivore, Meal Type: Breakfast, Ski...	['Caramel Breakfast Rolls', ['2 (1 lb.) loave...	Create a recipe based on the following
3	Diet Type: Omnivore, Meal Type: Breakfast, Ski...	['Egg And Sausage Muffins', ['1/4 cup cream ...	Create a recipe based on the following
4	Diet Type: Omnivore, Meal Type: Dinner, Skill ...	['Taco Soup', ['2 lb. lean ground beef', "1 s...	Create a recipe based on the following

Figura 3.7: Ejemplo de la estructura del conjunto de datos adaptado para el escenario 2.

Finalmente, tanto el conjunto de datos para el escenario 1 como el del escenario 2 se almacenarán en Hugging Face para luego ser utilizados en el fine-tuning del modelo.

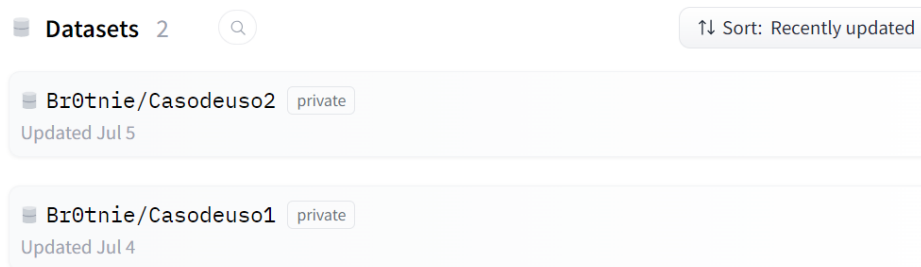


Figura 3.8: Conjuntos de datos adaptados para los escenarios definidos, alojados en Hugging Face.

Ambos conjuntos de datos fueron divididos en 4500 registros para el entrenamiento y 500 registros para evaluar el modelo.

3.5 Interfaz

La interfaz gráfica del sistema fue construida con Streamlit [25], una herramienta que destaca por su simplicidad en la creación de aplicaciones web en Python [26]. Ejecutada localmente, esta interfaz ofrece a los usuarios una forma directa y accesible de generar recetas para los escenarios definidos en este proyecto.

La aplicación se comunica con los modelos específicos para cada escenario a través de la Inference API de Hugging Face [27], donde fueron subidos tras el proceso de ajuste con sus respectivos conjuntos de datos.

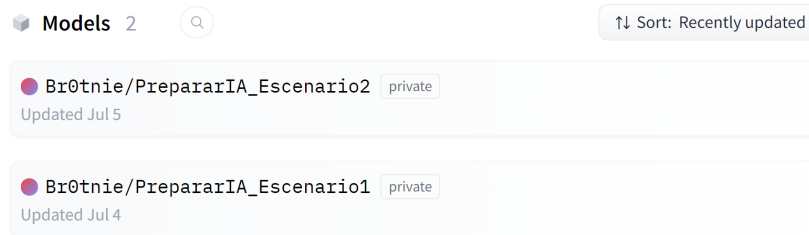


Figura 3.9: Modelos ajustados para generar recetas según los escenarios definidos, alojados en Hugging Face.

El diseño de la página de inicio es sencillo y cuenta con dos secciones principales, cada una asociada a un escenario específico. La primera sección permite la **generación de recetas a partir de ingredientes disponibles**, mientras que la segunda se enfoca en la **generación de recetas a partir de restricciones alimentarias**. Estas secciones están directamente vinculadas con los escenarios definidos en este proyecto a través de botones, lo que facilita que el usuario acceda rápidamente a la herramienta que necesita. Además, se incluye una advertencia sobre la responsabilidad en el uso de las recetas generadas, destacando que PrepararIA es una herramienta de apoyo y no garantiza resultados.

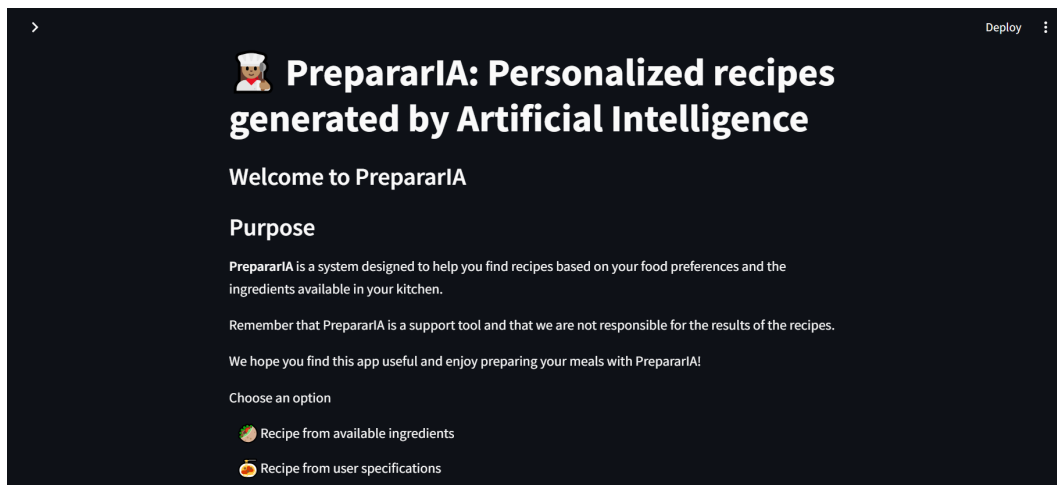


Figura 3.10: Página de inicio de la aplicación web PrepararIA.

En la figura 3.11 se muestra la página dedicada al escenario 1, donde el usuario selecciona la cantidad de ingredientes que desea utilizar e introduce los ingredientes correspondientes. Al hacer clic en el botón 'Generate recipe', la interfaz despliega la receta generada por el sistema, basada en la información proporcionada.

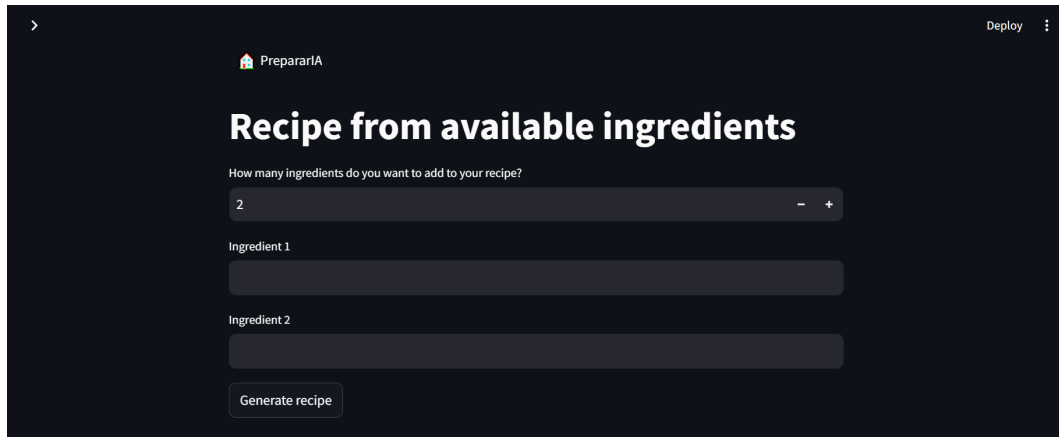
The screenshot shows a web application interface with a dark theme. At the top, there is a navigation bar with a home icon and the text 'PrepararIA', and a 'Deploy' button with a menu icon. The main heading is 'Recipe from available ingredients'. Below it, a prompt asks 'How many ingredients do you want to add to your recipe?'. A numeric input field contains the value '2', with minus and plus buttons on either side. Below this are two text input fields labeled 'Ingredient 1' and 'Ingredient 2'. At the bottom is a 'Generate recipe' button.

Figura 3.11: Página web del escenario 1: Recetas basadas en ingredientes disponibles.

La figura 3.12 muestra la página dedicada al escenario 2, donde se encuentran tres listas desplegables que permiten al usuario especificar los requerimientos para generar la receta. La primera lista permite seleccionar el tipo de dieta o preferencia alimentaria del usuario, la segunda define el tipo de platillo que se desea preparar, y la tercera permite indicar el nivel de habilidad culinaria, ya sea el que posee el usuario o el que desea aplicar en la receta.

The screenshot shows a web application interface with a dark theme. At the top, there is a navigation bar with a home icon and the text 'PrepararIA', and a 'Deploy' button with a menu icon. The main heading is 'Recipe based on food preferences and culinary skills.'. Below it, a prompt asks 'Specify your requirements.'. There are three dropdown menus: 'Diet Type' with 'Omnivore' selected, 'Meal Type' with 'Breakfast' selected, and 'Skill Level' with 'Beginner' selected. At the bottom is a 'Generate recipe' button.

Figura 3.12: Página web del escenario 2: Recetas basadas en preferencias y restricciones alimentarias.

4 | Resultados

En este capítulo se presentan los resultados del sistema, abarcando tanto su funcionamiento como su evaluación cualitativa y cuantitativa.

4.1 Operación del Sistema

La figura 4.1 muestra el sistema diseñado para el escenario 1 en ejecución.

The screenshot displays the 'PrepararIA' web application interface. At the top, the title 'Recipe from available ingredients' is prominently displayed. Below this, a prompt asks 'How many ingredients do you want to add to your recipe?' with a numeric input field set to '3'. Three ingredient input fields follow, labeled 'Ingredient 1', 'Ingredient 2', and 'Ingredient 3', containing the text 'basil', 'tomato', and 'eggs' respectively. A red 'Generate recipe' button is positioned below these inputs. The generated recipe is titled 'Basil Tomato Scrambled Eggs'. It includes an 'Ingredients:' section with a bulleted list: '3 large eggs', '1 medium tomato, diced', 'A handful of fresh basil leaves, roughly chopped', 'Salt and pepper to taste', and '1 tablespoon olive oil or butter'. The 'Instructions:' section follows with a numbered list of nine steps, detailing the process from cracking eggs to serving.

PrepararIA

Recipe from available ingredients

How many ingredients do you want to add to your recipe?

3

Ingredient 1

basil

Ingredient 2

tomato

Ingredient 3

eggs

Generate recipe

Basil Tomato Scrambled Eggs

Ingredients:

- 3 large eggs
- 1 medium tomato, diced
- A handful of fresh basil leaves, roughly chopped
- Salt and pepper to taste
- 1 tablespoon olive oil or butter

Instructions:

1. Crack the eggs into a bowl, season with a pinch of salt and pepper, and beat well.
2. Heat the olive oil or butter in a non-stick skillet over medium heat.
3. Add the diced tomato to the skillet and cook for 2-3 minutes until slightly softened.
4. Pour the beaten eggs into the skillet.
5. Let the eggs cook undisturbed for a few seconds until they begin to set around the edges.
6. Gently stir and fold the eggs with a spatula, allowing the uncooked parts to flow to the edges.
7. When the eggs are mostly set (still slightly runny), add the chopped basil.
8. Continue to cook, folding gently, until the eggs are fully cooked but still soft.
9. Remove from heat and serve immediately.

Figura 4.1: Vista de la operación del sistema en el escenario 1.

La figura 4.2 muestra el sistema diseñado para el escenario 2 en ejecución.

The screenshot shows a web application titled "PrepararIA" with a dark theme. The main heading is "Recipe based on food preferences and culinary skills." Below this, a section titled "Specify your requirements." contains three dropdown menus: "Diet Type" set to "Vegan", "Meal Type" set to "Dessert", and "Skill Level" set to "Intermediate". A red "Generate recipe" button is positioned below these menus. The generated recipe is titled "Vegan Chocolate Avocado Mousse". It includes a list of ingredients: 2 ripe avocados, 1/4 cup unsweetened cocoa powder, 1/4 cup maple syrup, 1/4 cup almond milk (or any plant-based milk), 1 teaspoon vanilla extract, a pinch of salt, and fresh berries or chopped nuts for garnish (optional). The instructions are a numbered list of seven steps: 1. Cut the avocados in half, remove the pit, and scoop the flesh into a blender or food processor. 2. Add the cocoa powder, maple syrup, almond milk, vanilla extract, and a pinch of salt. 3. Blend until the mixture is smooth and creamy, scraping down the sides as needed. 4. Taste and adjust sweetness if necessary by adding more maple syrup. 5. Spoon the mousse into serving bowls or glasses. 6. Refrigerate for at least 30 minutes to allow it to set. 7. Garnish with fresh berries or chopped nuts before serving if desired.

Figura 4.2: Vista de la operación del sistema en el escenario 2.

4.2 Evaluación del Sistema

4.2.1 Rouge: Recall-Oriented Understudy for Gisting Evaluation

ROUGE [11] es una métrica utilizada para evaluar la calidad del contenido generado por LLMs, inicialmente concebida para tareas de resumen automático de texto. Compara el contenido generado con el de referencia, midiendo la coincidencia de n-gramas. Sus submétricas, como ROUGE-1 (coincidencia de palabras individuales), ROUGE-2 (pares de palabras) y ROUGE-L (subsecuencias más largas), permiten cuantificar la relación entre ambos. ROUGE también permite calcular *precision*, *recall* y *F1-Score*, proporcionando una visión más detallada del rendimiento del modelo.

Este método de evaluación se considera relevante para el proyecto, ya que permite medir la calidad de las recetas generadas mediante la comparación con recetas de referencia. Proporcionando una evaluación objetiva de la precisión y cobertura del modelo.

El cálculo de las métricas ROUGE se realizó utilizando la biblioteca datasets de Hugging Face, como se ilustra en la figura 4.3. Esta biblioteca permite cargar la métrica y comparar el texto generado por el modelo con el texto de referencia.

```
from datasets import load_metric

rouge = load_metric("rouge")
predictions = lista_texto_generado
references = lista_texto_referencia
rouge.compute(predictions=predictions, references=references)
```

Figura 4.3: Código utilizado para el cálculo de las métricas ROUGE.

Cada métrica devuelve tres niveles de confianza: *low*, *mid* y *high*, que representan el límite inferior, el valor central y el límite superior, respectivamente. Se determinó considerar solo *mid* ya que refleja el rendimiento promedio del modelo.

Para evaluar el modelo ajustado para el **escenario 1**, se utilizó el set de datos de prueba, generando nuevas recetas a partir de los ingredientes disponibles, las cuales se consideraron como el texto generado. Las recetas originales del dataset, asociadas a esos mismos ingredientes, se tomaron como las recetas de referencia. En la figura 4.4 se puede visibilizar un ejemplo de este proceso.

ROUGE evaluó la coincidencia entre ambos textos. Los resultados detallados de esta evaluación se pueden visualizar en el cuadro 4.1.

```
entrada = ["butter", "sugar", "vanilla", "eggs", "coconut", "pie crust"]
texto_generado = "Coconut Cream Pie, [1 c. butter, 1 c. sugar, 1 tsp. vanilla, 4 eggs, 1 c. coconut], [Cream butter and sugar., Add vanilla and eggs., Add coconut., Prebake pie crust for 10-15 minutes at 375 °F (optional)., Pour mixture into pie crust., Bake at 200 °C for 20 minutes.]"
referencia = "Coconut Pie, [1/4 c. butter, 1 c. sugar, 1/4 c. Karo, 1 tsp. vanilla, 3 beaten eggs, 1 c. coconut], [Cream together butter, sugar and Karo., Add vanilla, eggs and coconut., Pour into pie crust., Bake at 375 °C for 45 to 50 minutes.]"
```

Figura 4.4: Ejemplo del texto generado y texto de referencia usados para evaluar el modelo del escenario 1.

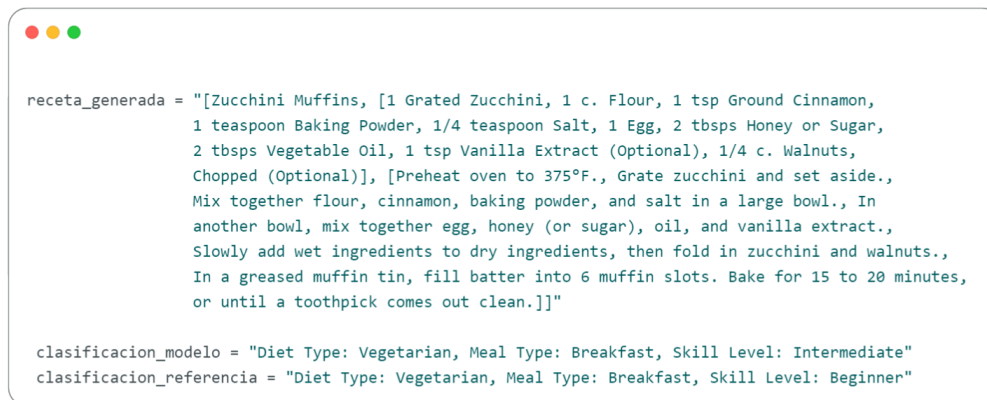
Métrica	Precision	Recall	F1-Score
ROUGE-1	0.72	0.7826	0.75
ROUGE-2	0.4082	0.4444	0.4255
ROUGE-L	0.62	0.6739	0.6458
ROUGE-Lsum	0.62	0.6739	0.6458

Cuadro 4.1: Evaluación del modelo ajustado para el escenario 1 mediante las métricas ROUGE.

Los resultados observados en el cuadro 4.1 permiten interpretar lo siguiente:

- **ROUGE-1:** Se observa una alta relación entre las recetas generadas y las recetas de referencia, lo que indica que los ingredientes principales aparecen consistentemente en ambas.
- **ROUGE-2:** Aunque el modelo logra incluir los ingredientes clave, presenta dificultades en mantener la secuencia exacta de los pasos o combinaciones de ingredientes entre la receta generada y la de referencia.
- **ROUGE-L y ROUGE-Lsum:** Las recetas generadas muestran una coherencia estructural aceptable con las recetas de referencia, lo que sugiere que el modelo es capaz de generar recetas con una secuencia lógica similar a la de las recetas de referencia.

Para evaluar el modelo en el **escenario 2**, donde las recetas se generan a partir de especificaciones del usuario, se adoptó un enfoque alternativo. Dado que el sistema puede generar múltiples recetas a partir de una misma especificación, en lugar de comparar directamente las recetas, se decidió comparar sus clasificaciones. Las recetas generadas a partir del conjunto de datos de prueba fueron clasificadas por el modelo gpt-3.5-turbo, para luego utilizar la clasificación generada como referencia para el análisis comparativo. A continuación, se aplicó la métrica ROUGE para medir la coincidencia entre las especificaciones originales del conjunto de datos de prueba y las clasificaciones obtenidas. La figura 4.5 ilustra un ejemplo de cómo se establecen la referencia y el texto generado.



```

receta_generada = "[Zucchini Muffins, [1 Grated Zucchini, 1 c. Flour, 1 tsp Ground Cinnamon,
1 teaspoon Baking Powder, 1/4 teaspoon Salt, 1 Egg, 2 tbsps Honey or Sugar,
2 tbsps Vegetable Oil, 1 tsp Vanilla Extract (Optional), 1/4 c. Walnuts,
Chopped (Optional)], [Preheat oven to 375°F., Grate zucchini and set aside.,
Mix together flour, cinnamon, baking powder, and salt in a large bowl., In
another bowl, mix together egg, honey (or sugar), oil, and vanilla extract.,
Slowly add wet ingredients to dry ingredients, then fold in zucchini and walnuts.,
In a greased muffin tin, fill batter into 6 muffin slots. Bake for 15 to 20 minutes,
or until a toothpick comes out clean.]]"

clasificacion_modelo = "Diet Type: Vegetarian, Meal Type: Breakfast, Skill Level: Intermediate"
clasificacion_referencia = "Diet Type: Vegetarian, Meal Type: Breakfast, Skill Level: Beginner"

```

Figura 4.5: Ejemplo del texto generado y texto de referencia usados para evaluar el modelo del escenario 2.

Métrica	Precision	Recall	F1-Score
ROUGE-1	0.89	0.89	0.89
ROUGE-2	0.76	0.76	0.76
ROUGE-L	0.89	0.89	0.89
ROUGE-Lsum	0.89	0.89	0.89

Cuadro 4.2: Evaluación del modelo ajustado para el escenario 2 mediante las métricas ROUGE.

En este escenario, se optó por considerar únicamente la métrica ROUGE-1 para evaluar el modelo. ROUGE-1 mide la coincidencia de elementos individuales entre las especificaciones del usuario y las clasificaciones generadas por el modelo GPT. Dado que estas especificaciones consisten en etiquetas independientes, como el tipo de dieta, tipo de plato y nivel de dificultad, la coincidencia a nivel de unigramas refleja si el modelo generó una receta adecuada a las restricciones del usuario. Esto se verifica al comparar la coincidencia con la clasificación generada por un modelo GPT avanzado, que actúa como una referencia consistente.

Considerando lo anterior, el valor presentado por ROUGE-1 en el cuadro 4.2 sugiere una alta correspondencia entre las especificaciones originales del usuario y las clasificaciones obtenidas del modelo GPT, lo que indica que el modelo ajustado es capaz de generar recetas adecuadas y alineadas con dichas especificaciones.

4.2.2 Percepción del usuario

En colaboración con el Laboratorio UX de la Universidad Técnica Federico Santa María, se realizó una evaluación cualitativa con 10 usuarios para medir la usabilidad y el rendimiento del sistema en los dos escenarios propuestos. Los participantes en esta evaluación fueron estudiantes universitarios, comprendidos en un rango de edad entre 18 y 25 años, capaces de entender recetas en inglés y que cocinan con una frecuencia de 2 a 4 veces por semana.

Los resultados obtenidos fueron mayormente positivos y se destacan a continuación:

Interfaz

- **Facilidad de navegación por la aplicación:** Según lo visualizado en la figura 4.6, la experiencia de navegación fue considerada intuitiva y fácil de usar por todos los usuarios, gracias a la simplicidad de la aplicación. Sin embargo, se sugirió mejorar la presentación visual de ciertos elementos gráficos, como los botones en la página de inicio, que algunos usuarios confundieron con texto debido a su diseño poco destacado.

Del 1 al 5, ¿qué tan fácil te resultó navegar por la aplicación?

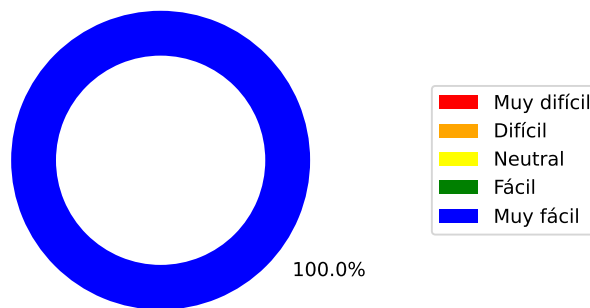


Figura 4.6: Gráfico de la percepción del usuario sobre la navegación en la aplicación.

Escenario 1: Recetas Basadas en Ingredientes Disponibles

- **Adecuación de la receta a los ingredientes:** Según lo visualizado en la figura 4.7, los usuarios interpretaron que las recetas generadas en el escenario 1, se ajustaban bien a los ingredientes proporcionados. No obstante, algunos comentaron que las proporciones eran demasiado grandes, sugiriendo que se debería implementar una opción para ajustar las porciones de las recetas generadas.

Del 1 al 5, ¿qué tan adecuada está la receta generada con los ingredientes que proporcionaste en la primera tarea?

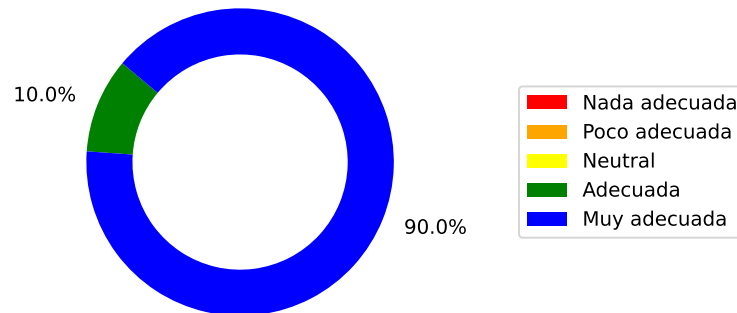


Figura 4.7: Gráfico de la percepción del usuario sobre la receta generada en el escenario 1.

Escenario 2: Recetas Basadas en Preferencias y Restricciones Alimentarias

- **Adecuación de la receta a las restricciones alimenticias:** Según lo visualizado en la figura 4.8, los usuarios consideraron que el sistema responde adecuadamente a las restricciones alimentarias especificadas. Algunos, sin embargo, sugirieron limitar la cantidad de ingredientes en las recetas, ya que en ciertas ocasiones el sistema generaba listas de ingredientes excesivamente largas.

Del 1 al 5, ¿qué tan adecuada está la receta generada con los ingredientes que proporcionaste en la segunda tarea?

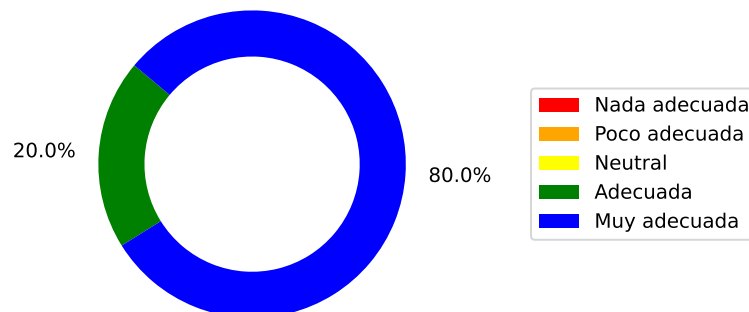


Figura 4.8: Gráfico de la percepción del usuario sobre la receta generada en el escenario 2.

Desempeño General

- **Claridad y fluidez de la estructura de las recetas:** Según lo visualizado en la figura 4.9, en su mayoría los usuarios percibieron las recetas como claras y secuencialmente correctas. Algunos usuarios señalaron que en ocasiones las instrucciones eran poco detalladas o incompletas.

Del 1 al 5, ¿qué tan clara y fluida te pareció la estructura de las recetas generadas (ingredientes e instrucciones)?

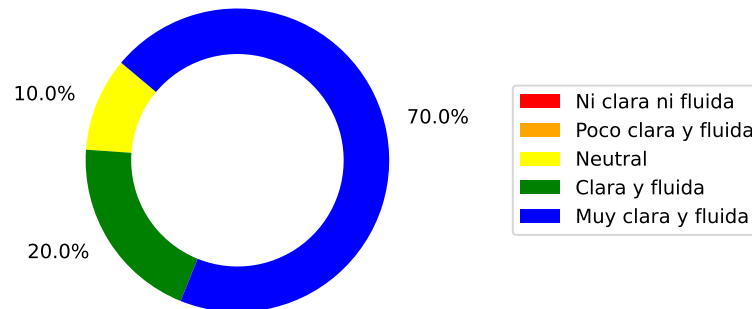


Figura 4.9: Gráfico de la percepción del usuario sobre la claridad y fluidez de las recetas generadas.

- **Creatividad de las recetas:** Según lo visualizado en la figura 4.10, los usuarios valoraron positivamente la creatividad del modelo en adaptar las recetas a sus preferencias y restricciones alimentarias. Sin embargo, algunos expresaron neutralidad, mencionando que algunas de las recetas generadas eran comunes y ya las habían visto antes en alguna otra plataforma.

Del 1 al 5, ¿qué tan creativa te pareció la manera en que la aplicación adaptó la receta a tus preferencias o restricciones alimenticias?

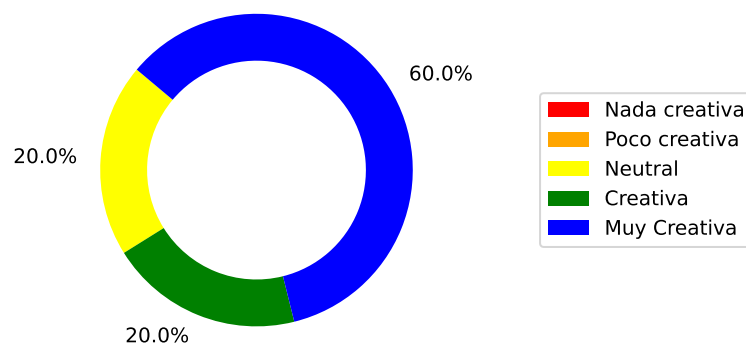


Figura 4.10: Gráfico de la percepción del usuario sobre la creatividad de las recetas generadas.

- **Probabilidad de prepararlas recetas generadas:** Según lo visualizado en la figura 4.11, las recetas generadas fueron consideradas no solo funcionales, sino también atractivas para la mayoría de los usuarios. Sin embargo, algunos mencionaron que, en las recetas generadas a partir de requerimientos de los usuarios, las listas de ingredientes de ciertas recetas eran demasiado largas, lo que reducía su probabilidad de prepararlas.

Del 1 al 5, ¿qué tan probable es que prepares alguna de las recetas que te generó la aplicación?

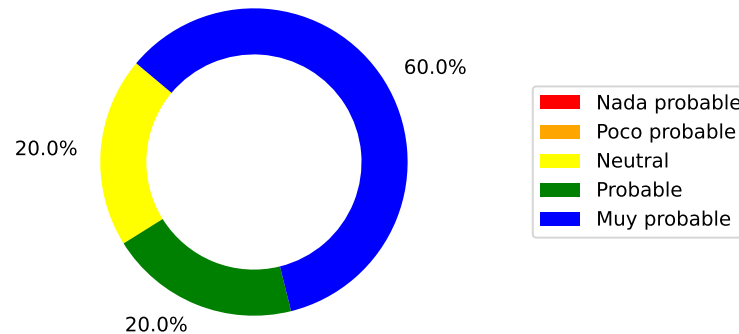


Figura 4.11: Gráfico de la percepción del usuario sobre la probabilidad de preparar las recetas generadas.

- **Tiempo de respuesta:** Según lo visualizado en la figura 4.12, el tiempo de respuesta del sistema al generar recetas fue valorado como aceptable y rápido por la mayoría de los usuarios. Varios mencionaron que, aunque la aplicación era rápida, no era una respuesta instantánea, lo que explica las variaciones en las calificaciones.

Del 1 al 5, ¿cómo evaluarías el tiempo de respuesta de la aplicación al generar las recetas?

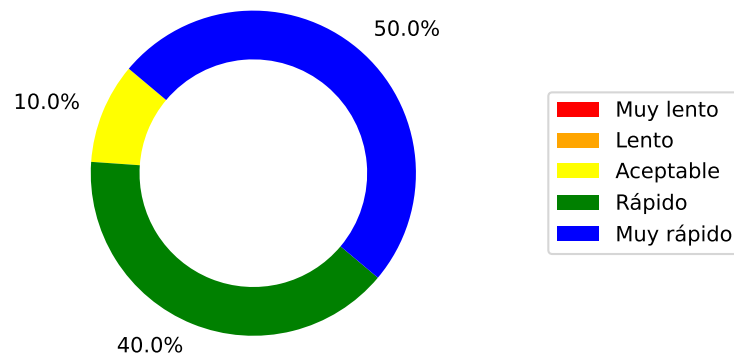


Figura 4.12: Gráfico de la percepción del usuario sobre el tiempo de respuesta del sistema.

- **Uso de la aplicación:** Según lo visualizado en la figura 4.13, todos los usuarios indicaron que utilizarían la aplicación, lo que refleja una fuerte aceptación del sistema en su conjunto.



Figura 4.13: Gráfico que indica la disposición de los usuarios a usar la aplicación.

5 | Conclusión

El desarrollo de este proyecto ha resuelto con éxito el desafío de generar recetas al ajustar un LLM a partir de conjuntos de datos específicos adaptados para los escenarios propuestos. El uso de QLoRA permitió optimizar el proceso y lograr resultados de alta precisión, sin la necesidad de reentrenar el modelo completo y adaptándose a entornos de bajos recursos.

Respecto a los experimentos realizados, se observó que el sistema sugiere recetas adaptadas tanto a los ingredientes disponibles como a las preferencias alimentarias del usuario, obteniendo buenos resultados incluso utilizando un modelo considerado pequeño dentro del espectro de los LLMs. El sistema recibió una alta aprobación tanto cualitativa como cuantitativamente, resultando en un producto atractivo para la comunidad estudiantil.

Se puede inferir que el rendimiento del sistema mejoraría con la incorporación de datos adicionales que permitan un desarrollo más completo en el ámbito culinario, siendo especialmente recomendable incluir a profesionales del área gastronómica para mejorar la calidad de los datos y, por ende, de los modelos. Para trabajos futuros, se sugiere expandir el conjunto de datos, incluyendo información sobre porciones, alimentos preferidos y prohibidos para acotar el espacio de posibles soluciones, así como instrucciones de seguridad. Asimismo, sería relevante desarrollar una métrica o marco de evaluación específico para proyectos de generación de contenido personalizado, lo que permitiría una evaluación más precisa del desempeño y adaptación del modelo a diferentes necesidades y contextos.

Este trabajo no solo contribuye al campo de la generación de recetas personalizadas, sino que también sirve como base para el desarrollo de otros proyectos que involucren modelos de lenguaje en contextos específicos. La metodología y resultados obtenidos en este proyecto ofrecen un marco adaptable para futuras aplicaciones que busquen aprovechar el poder de los LLMs en tareas de generación de contenido adaptado a las necesidades del usuario.

Bibliografía

- [1] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, *et al.*, “Mistral 7b,” *arXiv preprint arXiv:2310.06825*, 2023.
- [2] H. H. Lee, K. Shu, P. Achananuparp, P. K. Prasetyo, Y. Liu, E.-P. Lim, and L. R. Varshney, “Recipegpt: Generative pre-training based cooking recipe generation and evaluation system,” in *Companion Proceedings of the Web Conference 2020*, pp. 181–184, 2020.
- [3] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, “Lora: Low-rank adaptation of large language models,” *arXiv preprint arXiv:2106.09685*, 2021.
- [4] I. Systems, “Countries with the least and most food waste.” <https://www.ifco.com/countries-with-the-least-and-most-food-waste/>, 2023. Accessed: 2024-07-29.
- [5] SafetyCulture, “Food waste statistics.” <https://training.safetyculture.com/statistics/food-waste-statistics/>, 2023. Accessed: 2024-07-29.
- [6] S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain, and J. Gao, “Large language models: A survey,” *arXiv preprint arXiv:2402.06196*, 2024.
- [7] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, “Qlora: Efficient finetuning of quantized llms,” *arXiv preprint arXiv:2305.14314*, 2023.
- [8] J. Li, T. Tang, W. X. Zhao, J.-Y. Nie, and J.-R. Wen, “Pre-trained language models for text generation: A survey,” *ACM Computing Surveys*, vol. 56, no. 9, pp. 1–39, 2024.
- [9] J. Devlin, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [10] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a method for automatic evaluation of machine translation,” in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318, 2002.
- [11] C.-Y. Lin, “ROUGE: A package for automatic evaluation of summaries,” in *Text Summarization Branches Out*, (Barcelona, Spain), pp. 74–81, Association for Computational Linguistics, July 2004.
- [12] T. B. Brown, “Language models are few-shot learners,” *arXiv preprint arXiv:2005.14165*, 2020.
- [13] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, “Language models are unsupervised multitask learners,”

- [14] J. Marin, A. Biswas, F. Offi, N. Hynes, A. Salvador, Y. Aytar, I. Weber, and A. Torralba, “Recipe1m+: A dataset for learning cross-modal embeddings for cooking recipes and food images,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019.
- [15] M. S. Razzaq, F. Maqbool, M. Ilyas, and H. Jabeen, “Evorecipes: A generative approach for evolving context-aware recipes,” *IEEE Access*, vol. 11, pp. 74148–74164, 2023.
- [16] P. Kansaksiri, P. Panomkhet, and N. Tantisuwichwong, “Smart cuisine: Generative recipe chatgpt powered nutrition assistance for sustainable cooking,” *Procedia Computer Science*, vol. 225, pp. 2028–2036, 2023. 27th International Conference on Knowledge Based and Intelligent Information and Engineering Systems (KES 2023).
- [17] V. Lialin, V. Deshpande, and A. Rumshisky, “Scaling down to scale up: A guide to parameter-efficient fine-tuning,” *arXiv preprint arXiv:2303.15647*, 2023.
- [18] A. Vaswani, “Attention is all you need,” *arXiv preprint arXiv:1706.03762*, 2017.
- [19] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai, “Gqa: Training generalized multi-query transformer models from multi-head checkpoints,” *arXiv preprint arXiv:2305.13245*, 2023.
- [20] R. Child, S. Gray, A. Radford, and I. Sutskever, “Generating long sequences with sparse transformers,” *arXiv preprint arXiv:1904.10509*, 2019.
- [21] Hugging Face, “Unsloth: Optimizing llm training.” <https://huggingface.co/unsloth>, 2023. Accessed: 2024-08-05.
- [22] H. Face, “Recipenlg dataset.” https://huggingface.co/datasets/mbien/recipe_nlg, 2023. Accessed: 2024-07-29.
- [23] J. Reback *et al.*, “pandas: powerful python data analysis toolkit.” <https://pandas.pydata.org/>, 2020.
- [24] OpenAI, “Gpt-3.5-turbo documentation.” <https://platform.openai.com/docs/models/gpt-3-5-turbo>, 2024. Accessed: 2024-07-29.
- [25] I. Streamlit, “Streamlit: A faster way to build and share data apps.” <https://streamlit.io>, 2021. Accessed: 2024-08-22.
- [26] P. S. Foundation, “Python language reference, version 3.x.” <https://www.python.org/>, 2023. Accessed: 2024-08-22.
- [27] H. Face, “Hugging face inference api.” <https://huggingface.co/inference-api>, 2023. Accessed: 2024-08-22.