



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Aplicación de modelos de lenguaje para análisis de encuestas

Grado en Ingeniería Mecánica

ALUMNO: Carlos Argos Rivero

DIRECTORA: Milagros Fernández Gavilanes

CURSO ACADÉMICO: 2024-2025

Universida_{de}Vigo



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Aplicación de modelos de lenguaje para análisis de encuestas

Grado en Ingeniería Mecánica
Intensificación en Tecnología Naval
Cuerpo General

Universida_deVigo

RESUMEN

Este Trabajo de Fin de Grado tiene como objetivo el diseño e implementación de un sistema automático capaz de procesar y analizar encuestas en papel, con el fin de reducir la carga de trabajo de los encuestadores mediante la automatización del tratamiento y presentación de sus resultados. Para ello, se han empleado modelos extensos de lenguaje, de código abierto, como herramienta para el análisis e interpretación de datos textuales.

El proyecto desarrollado abarca todo el flujo de trabajo, desde la digitalización de encuestas mediante OCR hasta la elaboración de informes estructurados en formato PDF, incluyendo tablas, gráficas y redacción automatizada de conclusiones, mediante la integración de herramientas de Python como Matplotlib o ReportLab. El sistema se ha diseñado para ejecutarse de manera *offline*, utilizando únicamente software libre y modelos accesibles como Llama3 y DeepSeek, adaptados a las limitaciones de hardware disponibles.

Los resultados obtenidos muestran un funcionamiento eficaz en la extracción y análisis de datos textuales, así como en la creación de informes finales. Sin embargo, los modelos de lenguaje empleados no han resultado válidos para el reconocimiento de respuestas seleccionadas inicialmente por el encuestado, por lo que se han desarrollado soluciones adicionales. El sistema cumple con éxito su objetivo principal, quedando preparado para incorporar mejoras futuras.

PALABRAS CLAVE

Procesamiento de texto, LLM, PLN, encuesta, análisis de resultados

AGRADECIMIENTOS

Quisiera dedicar unas palabras de agradecimiento a quienes me han acompañado y apoyado a lo largo del desarrollo de este proyecto, y también durante toda mi etapa en la Escuela.

En primer lugar, a mi tutora, por su paciencia, dedicación y compromiso durante todo el proceso de tutorización. Su orientación ha sido clave para la finalización de este trabajo, y su apoyo ha estado presente hasta el último momento.

A mis colegas de la Escuela, por convertir los meses de trabajo en una experiencia mucho más llevadera. Gracias por haber estado siempre disponibles, por las ideas compartidas y por los ratos que nos han ayudado a desconectar y recuperar fuerzas.

Y, por supuesto, a mi familia y a Lucía, por ser el soporte incondicional que nunca ha fallado. Por haber sabido animarme en cada paso y darme el empujón necesario cuando hacía falta; y por demostrar que, incluso en la distancia, el apoyo de los que te quieren sigue siendo lo más importante.

Contenido

Índice de Figuras	4
Índice de Tablas	6
1 Introducción y objetivos.....	7
1.1 Introducción	7
1.2 Motivación	9
1.3 Objetivos	9
1.4 Estructura del trabajo	10
2 Estado del arte	12
2.1 Introducción a la Inteligencia Artificial	12
2.1.1 Comienzo de la Inteligencia Artificial	12
2.1.2 Antecedentes en estudio de teorías neuronales	13
2.1.3 Avances en las décadas posteriores.....	14
2.1.4 Siglo XXI: Aprendizaje profundo y explosión de la IA.....	15
2.1.5 Capacidades de la Inteligencia Artificial.....	15
2.2 Machine learning y tipos de aprendizaje	16
2.2.1 Aprendizaje supervisado	16
2.2.2 Aprendizaje no supervisado	18
2.2.3 Aprendizaje semisupervisado.....	20
2.2.4 Aprendizaje por refuerzo.....	21
2.2.5 Deep Learning y las redes neuronales	22
2.3 Desarrollo de los LLMs	23
2.3.1 Origen y desarrollo	23
2.3.2 Principios fundamentales de los LLMs	25
2.3.3 El desarrollo de los modelos a gran escala.....	26
2.3.4 Desafíos y limitaciones tecnológicas de los LLMs	28
2.4 Introducción al análisis de encuestas	29
2.4.1 Definición y propósitos	29
2.4.2 Aplicaciones en distintos ámbitos	29
2.4.3 Importancia del análisis automatizado	30
2.4.4 Métodos estadísticos y el uso de IA en el análisis	30
2.4.5 Tipos de encuestas.....	31
2.5 Reconocimiento óptico de caracteres y digitalización.....	32
2.5.1 Principios y funcionamiento del OCR	32
2.5.2 Aplicación del OCR en este trabajo	33

2.5.3 Modelos de OCR actuales	34
3 Desarrollo del trabajo.....	36
3.1 Entorno de trabajo y herramientas empleadas	36
3.1.1 Entornos de trabajo empleados	36
3.1.2 Lenguaje Python.....	40
3.1.3 Librería de python empleadas en el proyecto.....	40
3.1.4 Modelos LLM seleccionados	41
3.2 Arquitectura del proyecto	43
3.2.1 Bloque 1: conversión de información del PDF a formato TXT	46
3.2.2 Bloque 2: extracción de respuestas del encuestado empleando LLM.....	47
3.2.3 Bloque 3: conversión de resultados a CSV y tratamiento de datos.....	50
3.2.4 Bloque 4: creación de gráficas de los resultados.....	51
3.2.5 Bloque 5: análisis de resultados mediante LLM	52
3.2.6 Bloque 6: generación automática de informes en PDF	54
3.2.7 Códigos de implementación de los LLMs.....	55
4 Resultados obtenidos.....	57
4.1 Caso de uso: encuesta realizada en Fundamentos de Redes de Ordenadores.....	57
4.2 Resultados de la extracción de respuestas	59
4.2.1 Resultados de la conversión de PDF a PNG	59
4.2.2 Resultados del OCR.	60
4.2.3 Resultados del script creador de prompts.....	61
4.2.4 Resultados del análisis básico de texto por los LLMs	65
4.2.5 Resultados del análisis de encuestas por los LLMs	69
4.3 Resultados del procesamiento de datos	74
4.3.1 Resultados de la biblioteca pandas.....	74
4.3.2 Resultados de la biblioteca matplotlib	75
4.3.3 Resultados del script creador de prompts.....	77
4.3.4 Resultados del análisis de respuestas por los LLMs	78
4.4 Resultados de la generación de informes.....	81
5 Conclusiones y líneas futuras.....	87
5.1 Conclusiones.....	87
5.2 Líneas futuras.....	87
6 Bibliografía	89
Anexo I: Implicaciones Sociales, Económicas y Ambientales	96
Anexo II: Reflexiones Éticas y Sociales	97

Anexo III: Código Principal: Main.py	98
Anexo IV: Códigos De Bloques De La Arquitectura	100
A4.1 Bloque_1.py: conversión de información del PDF al formato TXT	100
A4.2 Bloque_2.py: extracción de encuestas del resultado empleando LLM	101
A4.3 Bloque_3.py: conversión de resultados a CSV y tratamiento de datos	102
A4.4 Bloque_4.py: creación de gráficas de resultados.....	104
A4.5 Bloque_5.py: análisis de resultados mediante LLM.....	106
A4.6 Bloque_6.py: generación automática de informes en PDF.....	109
Anexo V: Códigos De Módulos Auxiliares	112
A5.1 Aux_OCR.py: código auxiliar para convertir PDF en TXT	112
A5.2 Aux_prompt.py: código auxiliar para generar <i>prompts</i> automáticamente	112
A5.3 Aux_CSV.py: código auxiliar para generar tablas CSV.....	114
A5.4 Aux_graficas.py: código auxiliar para generar gráficas	117
A5.5 Aux_informe.py: código auxiliar para generar los informes en PDF	118
Anexo VI: Códigos De Implementación De LLM.....	124
A6.1 LLM_DeepSeek.py: código de implementación del modelo DeepSeek-R1 7B	124
A6.2 LLM_Llama3.py: código de implementación del modelo Llama-3 de Meta.....	124

ÍNDICE DE FIGURAS

Figura 2-1 Proyecto de verano de Dartmouth College en 1956 [6].	13
Figura 2-2 Interfaz del programa ELIZA [14].	14
Figura 2-3 Diferencia de uso entre clasificación y regresión [44].	17
Figura 2-4 Aprendizaje supervisado [45].	18
Figura 2-5 Aprendizaje no supervisado [48].	19
Figura 2-6 Aprendizaje semisupervisado [53].	20
Figura 2-7 Aprendizaje por refuerzo [56].	22
Figura 2-8 Capas de una red de neuronas [61].	23
Figura 3-1 Interfaz de GoogleColab (elaboración propia).	37
Figura 3-2 Menú de selección de entornos de ejecución (elaboración propia).	37
Figura 3-3 Interfaz de servidor de CUD mediante el terminal de Windows (elaboración propia).	38
Figura 3-4 Interfaz de desarrollo de PyCharm (elaboración propia).	39
Figura 3-5 Interfaz de FileZilla (elaboración propia).	39
Figura 3-6 Repositorio en Hugging Face de la versión de Llama 3 usada (elaboración propia).	42
Figura 3-7 Repositorio en Hugging Face de la versión de DeepSeek usada (elaboración propia).	43
Figura 3-8 Esquema de la arquitectura del proyecto (elaboración propia).	44
Figura 3-9 Desglose de encuestas de un grupo desde <i>Scanned_PDF</i> (elaboración propia).	45
Figura 3-10 Ficheros de código empleados (elaboración propia).	45
Figura 4-1 Imágenes producto de la conversión de PDF a PNG (elaboración propia).	60
Figura 4-2 Texto extraído de la encuesta original (elaboración propia).	60
Figura 4-3 Texto extraído de una encuesta del quinto curso (elaboración propia).	61
Figura 4-4 Pregunta simple a Llama3 cuantificado (elaboración propia).	65
Figura 4-5 Localización de errores en texto por Llama3 cuantificado (elaboración propia).	65
Figura 4-6 Respuesta formalizada por Llama3 cuantificado (elaboración propia).	66
Figura 4-7 Pregunta simple a Llama3 no cuantificado (elaboración propia).	66
Figura 4-8 Localización de errores en texto por Llama3 no cuantificado (elaboración propia).	67
Figura 4-9 Respuesta formalizada por Llama3 no cuantificado (elaboración propia).	67
Figura 4-10 Pregunta simple al modelo de DeepSeek (elaboración propia).	68
Figura 4-11 Localización de errores en texto por el modelo DeepSeek (elaboración propia).	68
Figura 4-12 Respuesta formalizada por el modelo DeepSeek (elaboración propia).	69
Figura 4-13 Archivo TXT con las respuestas de las encuestas de un grupo (elaboración propia).	70
Figura 4-14 Respuesta Llama3 con cuantificación (elaboración propia).	70
Figura 4-15 Respuesta Llama3 sin cuantificación (elaboración propia).	71
Figura 4-16 Razonamiento DeepSeek en análisis de encuestas (elaboración propia).	71

Figura 4-17 Respuesta del análisis de encuestas de DeepSeek (elaboración propia).	72
Figura 4-18 Respuesta de GhatGPT para el prompt propuesto (elaboración propia).	72
Figura 4-19 Respuesta de Claude para el prompt propuesto (elaboración propia).	73
Figura 4-20 Respuesta de DeepSeek online para el prompt propuesto (elaboración propia).	73
Figura 4-21 Respuesta de Gemini para el prompt propuesto (elaboración propia).	73
Figura 4-22 Generación de encabezado por el modelo DeepSeek (elaboración propia).	74
Figura 4-23 Sección de tabla de resultados del grupo 2 del curso 5 (elaboración propia).	74
Figura 4-24 Archivo TXT transición a CSV (elaboración propia).	75
Figura 4-25 Sección de tabla CSV recopilatorio de datos (elaboración propia).	75
Figura 4-26 Sección de tabla de transición actualizada con datos totales (elaboración propia).	76
Figura 4-27 Ejemplo de gráfica de barras (elaboración propia).	76
Figura 4-28 Figura 4 26 Ejemplo de gráfica circular o de pastel (elaboración propia).	77
Figura 4-29 Análisis resultados Llama3 cuantificado (elaboración propia).	78
Figura 4-30 Análisis resultados Llama3 no cuantificado (elaboración propia).	79
Figura 4-31 Análisis resultados del modelo DeepSeek (elaboración propia).	79
Figura 4-32 Conclusión informe Llama3 cuantificado (elaboración propia).	80
Figura 4-33 Conclusión informe Llama3 no cuantificado (elaboración propia).	80
Figura 4-34 Conclusión informe modelo DeepSeek (elaboración propia).	81
Figura 4-35 Encabezado informe PDF (elaboración propia).	81
Figura 4-36 Captura de la pregunta 5 del informe del quinto curso (elaboración propia).	82
Figura 4-37 Captura de la conclusión del informe del quinto curso (elaboración propia).	82
Figura 4-38 Primera cara informe quinto curso (elaboración propia).	83
Figura 4-39 Cara intermedia informe quinto curso (elaboración propia).	84
Figura 4-40 Última cara informe quinto curso (elaboración propia).	85

ÍNDICE DE TABLAS

Tabla 2-1 Comparación algoritmos OCR (elaboración propia).....	34
Tabla 3-1 Código <i>subprocess</i> en el código Main.py (elaboración propia)	46
Tabla 3-2 Función de conversión de PDF a TXT (elaboración propia).....	47
Tabla 3-3 Código generación de <i>prompt</i> para extraer encabezado (elaboración propia).	48
Tabla 3-4 Código de secuencia de generación de encabezado (elaboración propia).....	49
Tabla 3-5 Código de generación de <i>prompt</i> para extracción de respuestas (elaboración propia).	50
Tabla 3-6 Secuencia de generación de tablas CSV (elaboración propia).	51
Tabla 3-7 Código de generación de gráficas (elaboración propia).	52
Tabla 3-8 Código para crear <i>prompt</i> de generación de análisis de preguntas (elaboración propia).	53
Tabla 3-9 Código para crear <i>prompt</i> de generación de conclusión (elaboración propia).	54
Tabla 3-10 Creación de vector con nombres de asignaturas (elaboración propia).	55
Tabla 3-11 Función de llamada al modelo Llama3 (elaboración propia).	55
Tabla 3-12 Función de llamada al modelo de DeepSeek (elaboración propia).	56
Tabla 4-1 Encuesta repartida al alumnado (elaboración propia).	59
Tabla 4-2 <i>Prompt</i> generado para buscar las respuestas (elaboración propia).....	63
Tabla 4-3 <i>Prompt</i> para realizar el encabezado de las tablas (elaboración propia).....	64
Tabla 4-4 Comparación análisis básico de LLM (elaboración propia).....	69
Tabla 4-5 Estudio de viabilidad de los LLM para el proyecto (elaboración propia).	72
Tabla 4-6 <i>Prompt</i> generado para analizar los resultados de una pregunta (elaboración propia).	77
Tabla 4-7 <i>Prompt</i> generado para generar la conclusión (elaboración propia).	78

1 INTRODUCCIÓN Y OBJETIVOS

1.1 Introducción

La evolución del ser humano ha comprobado que esta se encuentra directamente condicionada por el desarrollo de avances tecnológicos, que, sin parecer al hombre actual ningún misterio, supusieron para los primeros hombres la diferenciación definitiva y necesaria para evolucionar al ser de razón y emocional al que nos hemos transformado [1]. Llegando a provocar que la tecnología, adelante procesos de la evolución [2].

Ya el hombre primitivo utilizaba las últimas herramientas en su poder para modernizar sus actividades con el fin de lograr una vida más tranquila y segura frente a las adversidades del mundo exterior. Es así como los sílex, la rueda e incluso el fuego revolucionaron las primeras sociedades humanas permitiendo emplear menos tiempo y recursos en actividades vitales y así poder emplear lo restante a satisfacer sus gustos y su creatividad.

Con el paso de los milenios los seres humanos fueron desarrollando incontables avances tecnológicos en numerosas áreas, especializándose, y creando el origen de la sociedad que conocemos. De esta forma, revoluciones científicas en campos como las matemáticas, la física, la química o la medicina, fueron haciendo más fuertes y avanzadas a las sociedades que conseguían nuevos hallazgos.

Y como aglomeración de todos estos factores: el instinto de supervivencia, el afán de superioridad y el reto personal. Un avance que supondría el origen de la cadena de avances más acelerada conocida hasta la fecha llegaría de la mano de un matemático inglés a mediados del siglo XX. Se trata de la primera máquina de computación mediante algoritmos.

Allan Turing creó el que se considera el primer ordenador de la historia, consiguiendo con ello la tan ansiada y necesaria ventaja sobre el enemigo: la Alemania nazi en la Segunda Guerra Mundial. Este fue el comienzo de una nueva lucha entre potencias mundiales para hacerse con los avances electrónicos y, posteriormente informáticos, que determinaran hacia qué lado se inclinaba la balanza del poder mundial [3].

Desde entonces, sufriendo numerosos altibajos por diversas tensiones políticas y sociales durante este periodo, las grandes potencias actuales continúan su carrera tecnológica para conseguir los mejores componentes con el software más optimizado, y capaz de hacer llegar a sus desarrolladores más lejos, más rápido y empleando menos recursos. Con el segundo milenio y sus avances en telecomunicaciones, la informática aprovechó el rápido desarrollo de la electrónica para no quedarse atrás y acompañar con nuevas capacidades que estuvieran a la altura.

El surgimiento de los ordenadores personales causó un cambio decisivo en toda la industria. Una nueva herramienta capaz de superar por mucha diferencia las capacidades de las personas en procesos

de cálculo y lógica. Poco después se extendería el uso de estas computadoras al entorno doméstico, coincidiendo con la creación de la red global de datos gracias a la aparición de Internet, que consiguió potenciar la expansión y utilización de la informática en todos los ámbitos de nuestro día a día.

La llegada del correo electrónico, las bases de datos online, herramientas de videoconferencia o de teletrabajo son ejemplos de cómo la informática ha revolucionado el concepto de trabajo en pocas décadas: mejora de técnicas productivas, automatización de procesos y análisis de riesgo mucho más elaborados, entre otros. Todos ellos tienen en común la ayuda que los ordenadores proveen a los humanos para su mejor rendimiento. Sin embargo, hasta la década de 2010 la posibilidad de usar la informática para desarrollar asistentes virtuales parecía una fantasía hollywoodense, y no habría sido posible sin el gran progreso en materia de aprendizaje profundo, o en adelante *Deep Learning* (DL), que dio la oportunidad al usuario corriente de empezar a trabajar asistido por lo último en tecnología: la Inteligencia Artificial (IA).

En la actualidad, una nueva etapa de la carrera tecnológica ha empezado, con un ritmo que parece inalcanzable para aquellos que no se encontraban ya en la vanguardia tecnológica [4]. Y, es que, el desarrollo de las IA que venía sucediendo a ritmos menores casi desde la aparición de la informática, ha logrado en el último lustro dar el salto que necesitaba para despuntar. Este acontecimiento trascendental no ha sido otro sino el surgimiento de las primeras IA Generativas (IAG). Particularmente, la llegada de ChatGPT¹ ha cambiado por completo la concepción de la IA.

A diferencia de los modelos anteriores a la aparición de la IAG, esta no se basa en modelos específicamente entrenados en una materia y capaces de predecir un posible resultado en base a los conocimientos adquiridos, sino que es capaz de analizar y razonar casi cualquier tema. Esto se debe gracias al desarrollo de los Modelos Extensos de Lenguaje, o LLM (*Large Language Models*). Su principal objetivo era conseguir la capacidad de entender texto y razonarlo para mejorarlo y reescribirlo. Sin embargo, han demostrado tener capacidades casi ilimitadas de consulta en casi cualquier campo, gracias al nutritivo entrenamiento que reciben de casi todos los ámbitos. De este modo, estamos viviendo una transición hacia un futuro en el que gran parte de los procesos y actividades que realiza el hombre podrían verse sustituidos por una máquina que los automatiza y completa en una cantidad de tiempo mucho menor.

Se ha convertido en objetivo fundamental de todas las instituciones y empresas la implementación en todo lo posible de herramientas basadas en IA, concretamente buscando aplicar el uso de las LLMs, con el fin de multiplicar la eficiencia y así mejorar resultados. Y no queda excluido de este cambio el ámbito universitario, donde la transformación digital ha sido casi completa, aunque todavía queda algún proceso puramente manual. Por ejemplo, el estudio de encuestas impresas, normalmente orientadas a la calidad de enseñanza o la valoración del profesorado, que todavía implican que estos profesores deban pasar a mano todos los resultados y analizar las respuestas comentadas de sus alumnos.

Otro ámbito que ha vivido una transición total a los cálculos automáticos es el análisis de datos y estadístico. Casi de manera natural, ya que un ordenador es una calculadora muy avanzada, estas ciencias adoptaron la computación como herramienta base de trabajo hace ya varias décadas, permitiendo realizar cálculos tediosos en cuestión de segundos y ayudando así a acelerar los descubrimientos en este campo. El análisis de datos ha vivido un desarrollo desenfrenado gracias a las notables mejoras de capacidades de cálculo y análisis en los diferentes modelos que han ido apareciendo hasta nuestros días.

Entendiendo estos avances podemos comprender que no resulta remota la idea de delegar el análisis de encuestas académicas a los nuevos ordenadores, que con la aparición de los LLMs han demostrado la mayor capacidad de adaptación y creación que se haya visto nunca, y así poder liberar de la carga laboral a profesores y personal educativo para poder dedicarlo a otras tareas como la investigación, sin

¹ ChatGPT de OpenAI disponible en: <https://openai.com/index/chatgpt/>. Accedido el 21/03/2024.

dejar de atender a la opinión del alumnado con la que es capaz de mejorar sus clases y metodología de enseñanza.

1.2 Motivación

La motivación para la elaboración de este trabajo se podría resumir en la ambición de automatizar un proceso que es recurrente, prolongado y bastante engorroso. La realización de encuestas para conocer la opinión o conocimientos de una población está a la orden del día, y cada vez es un recurso más recurrido. Es por esto por lo que numerosos estudiosos continúan investigando cómo modernizar y hacer más eficientes los métodos tradicionales. Sin embargo, por el momento, una parte notable de las personas que recurren a la elaboración y reparto de encuestas siguen prefiriendo métodos menos modernos y digitalizados. La encuesta en papel, como se estudia en el presente trabajo, sigue mostrando ciertas ventajas a las que algunos encuestadores no quieren renunciar. No por ello, los métodos más evolucionados, que recurren a plataformas digitales para obtener las respuestas de las encuestas, muestran numerosas ventajas a la hora del análisis, no solo en recursos y tiempo de empleo, sino en la calidad de los análisis.

De esta manera, surge la idea de este proyecto: conseguir mantener las ventajas de las encuestas a papel y las ventajas de las encuestas digitales. Para ello, se propone la generación de un software que sea capaz de recibir las encuestas de papel y proceder con el resto del proceso de análisis como si se hubieran recibido los resultados de manera digital. Además, aprovechando el creciente desarrollo de los LLMs en el área de análisis, estudiar la posibilidad de reemplazar al humano también en las tareas de análisis de resultados y presentación de un informe final, convirtiendo el método actual de análisis de encuestas en un nuevo procedimiento automatizado que solo requiera recibir encuestas en formato físico y devuelva informes digitales con los análisis pertinentes realizados.

Por otro lado, en cuanto a la motivación personal, siendo el Trabajo de Fin de Grado un proyecto que puede resultar inquietante por su extensión, surge la necesidad de aprovechar tantas horas de trabajo para inventar, generar e ingeniar algo verdaderamente útil. Que todo ese trabajo sea con un objetivo real y útil.

En mi caso, he comprobado de primera mano cómo la tecnología está fuertemente ligada con todos los ámbitos de la ingeniería. Prácticamente todas las áreas dependen de herramientas computacionales que han revolucionado la manera de diseñar y gestionar. Por ello, llevo desde primer curso creyendo que la realización de este proyecto sería la mejor de las oportunidades para acercarse al origen de todas estas herramientas: la informática.

Sumado a este hecho, el asombro que me ha generado los recientes avances en tecnologías de IA, brindando una oportunidad de oro para ponerse al día en todo lo relacionado con esta materia, y así conseguir permanecer a la vanguardia de lo que ya es la carrera tecnológica más acelerada que haya habido.

Recuperando el hilo inicial se puede entender que, al presentarse la oportunidad de hacer este trabajo, vi clara la oportunidad de aprender sobre los LLMs y conocer cómo gestionar sus recursos empleando programación en Python y otros recursos actuales como servidores externos con grandes capacidades GPU, como ofrecen instituciones como el CUD-ENM u online como la herramienta Google Colab, pudiendo crear un programa verdaderamente útil y con resultados palpables, y así cumplir con el objetivo de cualquier ingeniero.

1.3 Objetivos

Este proyecto nace de la propuesta de automatizar el procesamiento y análisis de encuestas en formato físico, con el fin de liberar de carga de trabajo a los encuestadores, que podrán recibir los resultados de manera eficiente y digital. El objetivo principal es conseguir un sistema que sea capaz de realizar todos los pasos inherentes al análisis de encuestas, requiriendo de los encuestadores únicamente

la carga en el sistema de las encuestas escaneadas, y ofreciendo como producto final un informe en PDF con el análisis y conclusiones de los resultados de la encuesta, apoyándose en ayudas gráficas.

El análisis de encuestas a papel comprende algunos procesos que habitualmente son llevados a cabo por humanos, al no contar con herramientas para hacerlo de forma digital. Entre estos pasos se encuentran, por ejemplo, la transcripción de las respuestas a formato digital o el análisis estadístico para extraer conclusiones de la encuesta. Con este proyecto se pretende automatizar también estos procesos.

Para ello se propone la utilización de LLMs debido a su gran capacidad de trabajo y razonamiento en casi cualquier ámbito. El objetivo es analizar si estos modelos, incorporados a un programa conductor, son capaces de suplir al encuestador en las tareas que hasta ahora eran exclusivamente humanas. Es requisito indispensable que todo el procesamiento sea independiente de cualquier servicio o API ubicado en un servidor ajeno online (considerando que en su mayoría son de pago por uso). Es decir, se pretende realizar un programa que ejecute los LLMs completamente en local empleando los recursos del propio sistema, sin depender de ninguna ayuda externa. Por estos motivos, la búsqueda de modelos permitidos se limita a aquellos de código abierto que puedan ser instalados en el propio equipo.

Esta decisión limita en gran medida el número y la calidad de los LLMs que pueden usarse para conseguir el resultado final. Como se mencionó en el apartado anterior, el desarrollo de los LLMs en los últimos cinco años ha sido frenético, y son numerosas las versiones que algunos desarrolladores ofrecen en abierto al público. Sin embargo, estos modelos gratuitos y libres van habitualmente varios pasos por detrás de sus homólogos online de pago, ofreciendo funciones más básicas y con menos capacidad. Por ello, es también un objetivo estudiar los LLMs de código abierto más modernos para estudiar su viabilidad en el proyecto, especialmente buscando alguno que ofrezca capacidad de trabajo con elementos distintos del texto, como pueden ser imágenes o documentos PDF.

1.4 Estructura del trabajo

Habiendo expuesto los objetivos de este proyecto se procede a explicar la estructura que sigue la presente memoria con el fin de entender con mayor claridad la presentación del estudio y el trabajo realizado, así como de los resultados obtenidos. El documento se compone de cinco capítulos, seguidos de una sección de bibliografía y de anexos complementarios. La organización de este TFG es la siguiente:

- Capítulo 1: Introducción y objetivos. En el primer capítulo de la memoria se presenta el proyecto realizado, haciendo una introducción al importante papel de los LLMs para la nueva era tecnológica. A continuación, se exponen los objetivos del proyecto explicando el proceso a seguir para obtener los resultados. Por último, se exponen las motivaciones de dicho trabajo y se explica la organización de la memoria.
- Capítulo 2: Estado del Arte. En el segundo capítulo se presenta el ámbito en el que se sitúa este trabajo. Los conceptos abordados en esta sección se pueden sintetizar en los siguientes puntos:
 - Introducción al origen de la Inteligencia Artificial y la revolución de esta en la nueva era tecnológica.
 - Introducción a los mecanismos de aprendizaje que siguen los modelos de lenguaje y sus aplicaciones en los distintos tipos de IA.
 - Aparición y desarrollo de los LLMs. Introducción a modelos de IA Generativa.
 - Introducción al análisis de encuestas. Estudio sobre los tipos de análisis de encuestas, tratando el procesamiento tradicional y al análisis de datos con IA.
 - Presentación de las herramientas de reconocimiento óptico de caracteres como enlace de los recursos físicos tradicionales y las versiones digitales.
- Capítulo 3: Desarrollo del trabajo. El tercer capítulo comienza con la explicación de los entornos y lenguaje empleados para el desarrollo del proyecto, seguido de las herramientas utilizadas para

conseguir que el proyecto realizara todos los cometidos. Después se presenta la arquitectura del sistema y su división en bloques de trabajo, que son explicados minuciosamente.

- Capítulo 4: Resultados obtenidos. En el cuarto capítulo se exponen los resultados obtenidos de los análisis realizados, así como el resultado del producto final y la verificación de los objetivos propuestos al inicio del proyecto.
- Capítulo 5: Conclusiones y líneas futuras. En el quinto capítulo se exponen las conclusiones tras el desarrollo del trabajo. Se analizan también las posibles líneas de continuación y mejora de este trabajo con el fin de seguir perfeccionando y acondicionando esta herramienta.
- Finalmente, se encuentra la bibliografía adjuntada donde se pueden encontrar todas las referencias empleadas en el desarrollo del trabajo y los anexos donde encontramos todo el código programado para este trabajo.

2 ESTADO DEL ARTE

2.1 Introducción a la Inteligencia Artificial

La IA ha evolucionado significativamente desde sus inicios, consolidándose como una disciplina clave en la ciencia y la tecnología modernas. Su desarrollo ha permitido la automatización de tareas complejas, impactando diversos sectores como la educación, la salud y la industria. En este apartado, se examinan los hitos históricos más relevantes de la IA y sus capacidades actuales, fundamentados en literatura científica.

2.1.1 Comienzo de la Inteligencia Artificial

El término "Inteligencia Artificial" fue acuñado en 1956 durante una reunión de verano de investigadores en IA en Dartmouth College, Estados Unidos. Celebrada en 1956, este evento es considerado el punto de partida oficial de la IA como disciplina científica. Fue organizada por John McCarthy, Marvin Minsky, Nathaniel Rochester y Claude Shannon, todos ellos presentes en la Figura 2-1, quienes propusieron que "cada aspecto del aprendizaje o cualquier otra característica de la inteligencia puede, en principio, ser descrito con tal precisión que una máquina puede simularlo" [5]. Los principales fundamentos establecidos en Dartmouth:

- Definición formal del campo de la IA: se introdujo el término "Inteligencia Artificial", marcando la distinción entre los métodos computacionales tradicionales y la posibilidad de que las máquinas pudieran emular procesos cognitivos humanos.
- Hipótesis de la IA: se postuló que la inteligencia puede ser representada simbólicamente y procesada mediante computadoras. Esta idea fundamentó los enfoques iniciales de la IA basada en reglas y lógica.
- Primeros modelos de aprendizaje automático: se exploraron las bases del aprendizaje automático, donde las máquinas podrían mejorar su desempeño a través de la experiencia sin una programación explícita para cada tarea.
- Sistemas basados en reglas y búsqueda heurística: se propuso el uso de sistemas expertos y heurísticas para resolver problemas complejos mediante la manipulación de símbolos y reglas lógicas.
- Fundación para la investigación posterior: a partir de esta conferencia, se establecieron los primeros programas que imitaban el razonamiento humano, como el programa Logic Theorist de Newell y Simon en 1956, considerado el primer sistema de IA.

La reunión de Dartmouth generó un gran optimismo sobre el desarrollo de la IA, con la creencia de que en poco tiempo las máquinas podrían resolver problemas complejos, comprender el lenguaje y aprender de manera autónoma. Aunque estas expectativas fueron demasiado ambiciosas debido a las limitaciones computacionales de la época, el evento logró consolidar la IA como una disciplina científica formal, impulsando su investigación y desarrollo en los años siguientes.



Figura 2-1 Proyecto de verano de Dartmouth College en 1956 [6].

Otro impacto clave fue el incremento en la financiación y la creación de laboratorios de investigación en IA en instituciones como el MIT y la Universidad de Stanford. Estos centros permitieron la experimentación con modelos simbólicos y sistemas expertos, sentando las bases para futuras innovaciones. Gracias a este impulso, investigadores como Marvin Minsky y John McCarthy avanzaron en el desarrollo de nuevos algoritmos y enfoques para la resolución de problemas mediante computación [7].

Finalmente, la conferencia incentivó la creación de herramientas especializadas en IA, como el lenguaje de programación LISP, diseñado por John McCarthy en 1958 para manipular datos simbólicos y facilitar el desarrollo de sistemas expertos. Este lenguaje se convirtió en el estándar en la investigación en IA y permitió avances significativos, como el programa Logic Theorist, que demostró la capacidad de las computadoras para resolver problemas lógicos de manera autónoma [8].

2.1.2 Antecedentes en estudio de teorías neuronales

En 1943, los investigadores Warren McCulloch y Walter Pitts publicaron el artículo titulado "A Logical Calculus of the Ideas Immanent in Nervous Activity" en el Bulletin of Mathematical Biophysics. En este trabajo, propusieron un modelo matemático del sistema nervioso, conceptualizando las neuronas como elementos lógicos binarios que podrían estar "activados" o "desactivados" [9]. Este enfoque permitió describir funciones cerebrales en términos abstractos y demostró que redes de neuronas interconectadas podían realizar cálculos complejos, sentando así las bases para futuras investigaciones en redes neuronales artificiales.

Este modelo teórico fue pionero en establecer una conexión entre la neurociencia y la lógica matemática, influenciando campos emergentes como la cibernética y la IA [10]. Aunque inicialmente no recibió amplia atención, posteriormente fue reconocido como un hito fundamental en la comprensión de los procesos cognitivos y en el desarrollo de modelos computacionales del cerebro.

La colaboración entre McCulloch (neurólogo y cibernético estadounidense) y Pitts (lógico estadounidense especializado en neurociencia computacional) combinó la experiencia en neurociencia y lógica matemática, respectivamente, resultando en una perspectiva innovadora sobre cómo las redes neuronales podrían emular procesos mentales complejos. Su trabajo continúa siendo una referencia esencial en el estudio de las redes neuronales y la IA.

2.1.3 Avances en las décadas posteriores

Durante las décadas de 1960 y 1970, la IA experimentó avances significativos que sentaron las bases para desarrollos posteriores en el campo. No obstante, también enfrentó desafíos significativos debido a limitaciones tecnológicas y expectativas no cumplidas, lo que llevó a periodos conocidos como "inviernos de la IA".

En 1965, Edward Feigenbaum y Joshua Lederberg iniciaron el proyecto DENDRAL en la Universidad de Stanford. Este fue uno de los primeros sistemas expertos diseñados para deducir estructuras moleculares de compuestos químicos a partir de datos espectrométricos. DENDRAL demostró cómo el conocimiento especializado podía incorporarse en programas informáticos para resolver problemas complejos en dominios específicos [11].

En la otra costa, Marvin Minsky y Seymour Papert publicaron "Perceptrons" en 1969 en el MIT de Boston, una obra que analizaba las capacidades y limitaciones de los perceptrones, un tipo temprano de red neuronal. Aunque señalaron restricciones en los modelos de una sola capa, su trabajo estimuló investigaciones posteriores en redes neuronales multicapa y algoritmos de aprendizaje más avanzados [12]. También en el MIT, Joseph Weizenbaum desarrolló en 1966 el programa ELIZA, como se ve en la Figura 2-2, uno de los primeros programas capaces de simular una conversación en lenguaje natural. ELIZA utilizaba patrones predefinidos para interactuar con los usuarios, sentando las bases para futuros desarrollos en procesamiento del lenguaje natural (PLN) y chatbots [13].

```

Welcome to
EEEEEE LL      IIII ZZZZZZZ AAAAA
EE      LL      II      ZZ  AA  AA
EEEEEE LL      II      ZZ  AAAAAAA
EE      LL      II      ZZ  AA  AA
EEEEEE LLLLLL IIII ZZZZZZZ AA  AA

Eliza is a mock Rogerian psychotherapist.
The original program was described by Joseph Weizenbaum in 1966.
This implementation by Norbert Landsteiner 2005.

ELIZA: Is something troubling you ?
YOU:   Men are all alike.
ELIZA: What is the connection, do you suppose ?
YOU:   They're always bugging us about something or other.
ELIZA: Can you think of a specific example ?

```

Figura 2-2 Interfaz del programa ELIZA [14].

Entre 1966 y 1972, el Instituto de Investigación de Stanford desarrolló a Shakey, considerado el primer robot móvil inteligente. Shakey integraba percepción, razonamiento y ejecución de acciones, pudiendo navegar en entornos estructurados y realizar tareas simples de forma autónoma [15].

A pesar de estos avances, la IA enfrentó desafíos significativos debido a expectativas no cumplidas y limitaciones tecnológicas de la época. La sobreestimación de las capacidades inmediatas de la IA y la falta de resultados prácticos condujeron a una disminución en la financiación y el interés, periodos conocidos como los "inviernos de la IA" [16] [17]. El resurgimiento de la IA en las décadas posteriores se atribuye en gran medida al incremento en la capacidad de cómputo y al acceso a grandes volúmenes de datos. Estos factores han sido cruciales para el desarrollo de técnicas avanzadas como el aprendizaje profundo [18].

Durante las décadas de 1980 y 1990, los sistemas expertos emergieron como una aplicación prominente de la IA, especialmente en campos como la medicina, las finanzas y la industria. Estos sistemas, basados en reglas predefinidas, buscaban emular el conocimiento y la experiencia de especialistas para resolver problemas complejos en áreas específicas. Un ejemplo destacado es el sistema MYCIN, desarrollado en la década de 1970, que asesoraba a médicos en el diagnóstico de infecciones bacterianas y la recomendación de tratamientos adecuados [19]. Otro caso es el sistema XCON, un sistema experto implementado por Digital Equipment Corporation para la configuración de sistemas informáticos personalizados, lo que resultó en un ahorro significativo para la empresa [20].

No obstante, la dependencia de estos sistemas en reglas rígidas limitaba su adaptabilidad. Carecían de la capacidad de aprender de nuevas experiencias, lo que dificultaba su aplicación en situaciones no contempladas originalmente [21]. Además, la actualización y el mantenimiento de la base de conocimientos requerían intervención humana constante, lo que podía ser costoso y laborioso [22]. Estas limitaciones evidenciaron la necesidad de desarrollar enfoques más flexibles y adaptativos en el campo de la IA [23].

2.1.4 Siglo XXI: Aprendizaje profundo y explosión de la IA

El siglo XXI ha sido testigo de una transformación significativa en el campo de la IA, especialmente con el auge del aprendizaje profundo en la década de 2010. Este avance fue posible gracias al incremento en la capacidad de cómputo y la disponibilidad de grandes volúmenes de datos, factores que permitieron el entrenamiento efectivo de redes neuronales profundas [24]. Estas redes han sido fundamentales en diversas aplicaciones, destacando las redes neuronales convolucionales (CNN) en tareas de visión artificial [25] y los modelos transformadores en el PLN [26].

El uso de CNNs permitió avances significativos en visión artificial, particularmente en la clasificación de imágenes mediante arquitecturas como AlexNet, que revolucionó el campo al demostrar el poder del aprendizaje profundo para el reconocimiento de patrones visuales [24]. Por otro lado, los modelos transformadores han redefinido el PLN con enfoques como BERT, que introdujo un preentrenamiento bidireccional capaz de mejorar la comprensión semántica de los textos [27].

Además, estudios recientes han destacado la evolución del aprendizaje profundo y su impacto en la IA moderna, mostrando cómo la combinación de grandes volúmenes de datos y arquitecturas optimizadas ha permitido avances sin precedentes en tareas cognitivas como la traducción automática, el análisis de sentimientos y la generación de texto [28].

2.1.5 Capacidades de la Inteligencia Artificial

Actualmente, la IA posee una amplia gama de capacidades que le permiten abordar tareas complejas en diversos dominios:

- **PLN:** La IA es capaz de comprender y generar lenguaje humano, facilitando interacciones más naturales entre humanos y máquinas. Esto se logra mediante modelos avanzados que analizan y producen texto coherente en múltiples idiomas [29].
- **Visión por computador:** La IA puede interpretar y procesar información visual, permitiendo aplicaciones como el reconocimiento de imágenes y la conducción autónoma. Estas capacidades se han potenciado gracias al desarrollo de CNNs [18].
- **Aprendizaje autónomo:** Los sistemas de IA pueden mejorar su rendimiento a través de la experiencia, adaptándose a nuevos datos sin intervención humana constante. Esto es esencial en áreas como los sistemas de recomendación y la detección de fraudes [18].
- **Automatización y optimización:** La IA se emplea para automatizar procesos y optimizar operaciones en sectores como la logística y la manufactura, incrementando la eficiencia y reduciendo costos [30].

La evolución constante de la IA continúa ampliando sus aplicaciones y capacidades, posicionándola como una herramienta indispensable en la sociedad contemporánea. Con la llegada de los LLMs la IA ha logrado grandes avances al haber conseguido potenciar todas las capacidades previamente comentadas. En PLN, los LLMs permiten a las máquinas comprender y generar lenguaje humano de manera más precisa, mejorando la interacción hombre-máquina [31]. En visión por computador, combinados con modelos multimodales, facilitan la interpretación de imágenes y su descripción en lenguaje natural [32]. Además, los LLMs contribuyen al aprendizaje autónomo al permitir que los sistemas adapten su comportamiento basándose en datos previos sin intervención humana constante [33]. En términos de automatización, estos modelos optimizan procesos empresariales al encargarse de tareas relacionadas con el lenguaje, como el análisis de datos y la generación de contenido, aumentando la eficiencia operativa [32].

2.2 Machine learning y tipos de aprendizaje

El aprendizaje automático o *Machine Learning* (ML) es la rama de la IA que estudia los métodos y algoritmos que permiten a las máquinas aprender a partir de datos sin ser explícitamente programadas. Su objetivo principal es desarrollar modelos que pueden identificar patrones en grandes volúmenes de datos (*Big Data*) [34], realizar predicciones y tomar decisiones en función de la información analizada. Además, los algoritmos de aprendizaje automático ajustan su comportamiento en función de los datos de entrada y la retroalimentación recibida, mejorando así su precisión y adaptabilidad. Dentro del aprendizaje automático, existen cuatro enfoques principales: aprendizaje supervisado, aprendizaje semisupervisado, aprendizaje no supervisado y aprendizaje por refuerzo, cada uno con características y aplicaciones específicas [35].

2.2.1 Aprendizaje supervisado

El aprendizaje supervisado es una técnica fundamental en el campo del aprendizaje automático que se basa en la utilización de conjuntos de datos etiquetados para entrenar modelos capaces de realizar predicciones o clasificaciones precisas [36]. Este enfoque implica que cada instancia del conjunto de datos de entrenamiento está asociada a una etiqueta o valor objetivo conocido, lo que permite al modelo aprender la relación entre las características de entrada y las salidas deseadas [37].

Existen dos tipos de aprendizaje supervisado, la clasificación y la regresión. Ambos enfoques comparten el objetivo de optimizar el rendimiento del modelo mediante el ajuste de parámetros y la minimización de errores, pero difieren en la naturaleza de sus salidas y en las técnicas específicas utilizadas para su implementación. La elección entre clasificación y regresión depende del tipo de datos y del problema que se desea resolver:

- Clasificación: es vital en el aprendizaje supervisado, donde el objetivo es asignar una etiqueta o categoría a cada entrada de datos en función de sus características. Los algoritmos de clasificación buscan identificar patrones en los datos que permitan diferenciar entre distintas clases. Algunos de los algoritmos más utilizados en clasificación incluyen:
 - Máquinas de vector soporte (SVM): estos algoritmos buscan encontrar el hiperplano que mejor separa las distintas clases en el espacio de características, maximizando el margen entre las clases [38].
 - Árboles de decisión: estructuras jerárquicas que dividen el espacio de características en subconjuntos más pequeños basándose en condiciones sobre las variables de entrada, facilitando la interpretación de las decisiones del modelo [39].
 - Redes neuronales artificiales: modelos inspirados en el funcionamiento del cerebro humano que consisten en capas de neuronas interconectadas capaces de aprender representaciones complejas de los datos [40].

Estos algoritmos pueden abordar diferentes tipos de problemas de clasificación. En el caso de la clasificación binaria, el modelo solo necesita distinguir entre dos posibles categorías. Por ejemplo, un sistema de diagnóstico médico podría clasificar si un paciente presenta una enfermedad o no (“enfermo” o “sano”). Por otro lado, en la clasificación multiclase, el modelo debe asignar cada dato de entrada a una de varias categorías posibles. Un ejemplo de esto sería un sistema de reconocimiento de especies de árboles a partir de imágenes de sus hojas, donde el algoritmo puede clasificar entre diferentes tipos de árboles. Además, existe la clasificación de etiquetas múltiples, en la que un mismo dato de entrada puede pertenecer a más de una categoría al mismo tiempo. Un ejemplo práctico de esto es un sistema de etiquetado automático de noticias, donde un artículo puede clasificarse simultáneamente en varias categorías, como “economía”, “política” y “tecnología”, dependiendo de su contenido.

- Regresión: es otra tarea esencial en el aprendizaje supervisado, enfocada en predecir valores continuos de una variable objetivo basándose en las variables de entrada. Los algoritmos de regresión buscan modelar la relación entre las variables independientes y la variable dependiente. Entre los métodos de regresión más comunes se encuentran:

- Regresión: existen regresión lineal y regresión polinómica. En la lineal se asume una relación lineal entre las variables independientes y la variable dependiente, ajustando una línea recta que minimice la suma de los errores cuadrados. En la polinómica se mejora la regresión lineal al considerar relaciones polinómicas entre las variables, permitiendo capturar relaciones más complejas [41] [42].
- Regresión con máquinas de vector soporte (SVR): una adaptación de las SVM para problemas de regresión, que busca encontrar una función que tenga al menos una desviación mínima de los datos de entrenamiento [43].

A diferencia de la clasificación, el rendimiento del algoritmo se evalúa midiendo la diferencia entre el valor predicho y el valor real, utilizando métricas como el error cuadrático medio o el error absoluto medio. Por ejemplo, un modelo de regresión podría utilizarse para predecir la temperatura de una ciudad en función de variables como la humedad, la presión atmosférica y la velocidad del viento. A diferencia de una clasificación (que podría indicar simplemente si hará “frío” o “calor”), este modelo proporcionaría un valor numérico específico, como 22.5 °C, y su precisión se mediría en función de qué tan cerca está esa predicción del valor real registrado.

La elección entre un modelo de clasificación o de regresión depende del tipo de variable objetivo que se desea predecir y del problema específico a resolver como se ve en la Figura 2-3. La clasificación se utiliza cuando los datos de salida pertenecen a un conjunto finito de categorías o clases y la regresión se utiliza cuando el objetivo es predecir un valor numérico continuo en función de variables de entrada.

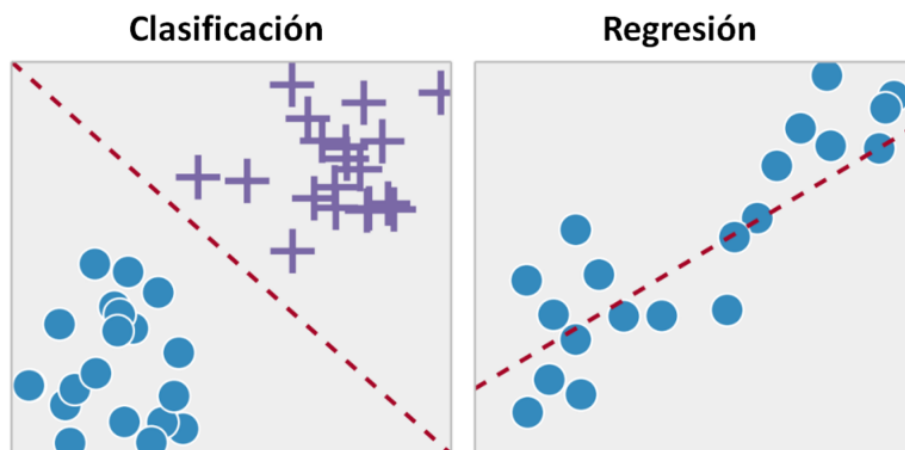


Figura 2-3 Diferencia de uso entre clasificación y regresión [44].

Para llevar a cabo este análisis y poder conseguir un modelo funcional, se lleva a cabo un proceso de tratamiento y estudio de datos, bien reflejado la Figura 2-4, que se puede resumir de manera general en los siguientes subprocesos:

1. Recolección y preparación de los datos: en primer lugar, se obtiene un conjunto de datos que contenga ejemplos en los que se conozca tanto la entrada como la salida correcta.
2. División de los datos en conjuntos de entrenamiento y prueba: una vez que se dispone del conjunto de datos, se divide en dos subconjuntos. El conjunto de entrenamiento se utiliza para enseñarle al modelo la relación entre las variables de entrada y la salida esperada, mientras que el conjunto de prueba se emplea para evaluar el rendimiento del modelo tras el entrenamiento.
3. Selección del algoritmo adecuado: dependiendo del tipo de problema a resolver, se elige un algoritmo específico. Si la salida del modelo corresponde a una categoría discreta, se trata de un problema de clasificación, si la salida es un valor numérico continuo, el problema es de regresión.
4. Entrenamiento del modelo: una vez seleccionado el algoritmo, el modelo se entrena utilizando los datos del conjunto de entrenamiento. Durante este proceso, el modelo analiza los datos y ajusta sus parámetros internos con el objetivo de minimizar la diferencia entre sus predicciones y los valores reales.

5. Evaluación del modelo: finalizado el entrenamiento, el modelo se evalúa con datos nuevos, correspondientes al conjunto de prueba, para determinar su precisión. Si las predicciones obtenidas son cercanas a los valores reales, el modelo se considera adecuado para su implementación. En caso contrario, pueden realizarse ajustes y repetir el entrenamiento.

Finalmente, el modelo entrenado se implementa en un entorno real para realizar predicciones con datos nuevos. Siguiendo el mismo ejemplo, un usuario que desee vender su vivienda podría ingresar sus características en el modelo y obtener una estimación del precio basada en ejemplos previos.

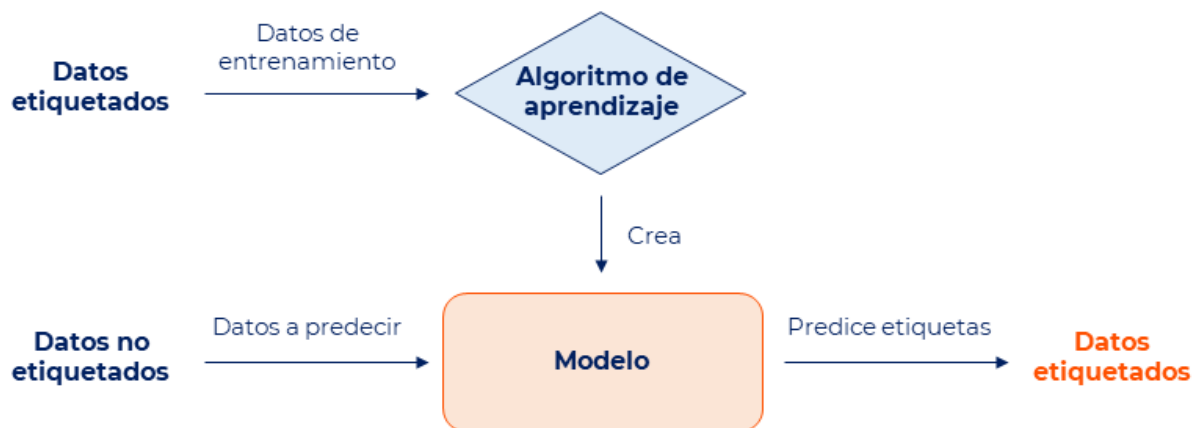


Figura 2-4 Aprendizaje supervisado [45].

Uno de sus principales desafíos del aprendizaje supervisado es la necesidad de contar con datos de entrenamiento bien estructurados y representativos, lo que requiere una recopilación y procesamiento exhaustivos. Encontramos aprendizaje supervisado en múltiples aplicaciones que forman parte de la vida cotidiana: en sistemas de detección de spam, asistentes virtuales, reconocimiento facial, predicción del mercado financiero y muchas otras tecnologías. Uno de los usos más relevantes del aprendizaje supervisado se encuentra en los sistemas de detección de fraudes bancarios [46]. Instituciones financieras como Visa, MasterCard y PayPal emplean modelos de clasificación supervisada para identificar transacciones sospechosas. Estos sistemas se entrenan con conjuntos de datos históricos que contienen tanto transacciones legítimas como fraudulentas, lo que permite que el modelo aprenda a reconocer patrones de comportamiento anómalos. Gracias a esta tecnología, los bancos pueden reducir significativamente los fraudes y alertar a los clientes ante posibles actividades inusuales.

2.2.2 Aprendizaje no supervisado

El aprendizaje no supervisado es una técnica de aprendizaje automático en la que un modelo analiza datos sin disponer de etiquetas o valores de referencia. A diferencia del aprendizaje supervisado, donde los algoritmos aprenden a partir de ejemplos etiquetados con una salida conocida, en el aprendizaje no supervisado el modelo debe encontrar patrones, relaciones o estructuras ocultas en los datos sin intervención humana directa. Este enfoque se aplica en situaciones donde no es viable o práctico etiquetar grandes volúmenes de datos, permitiendo que el algoritmo descubra tendencias y agrupaciones de manera autónoma. En lugar de recibir pares de entrada y salida, el modelo trabaja exclusivamente con los datos de entrada, explorando similitudes, diferencias y asociaciones entre los elementos del conjunto de datos [47].

La Figura 2-5; **Error! No se encuentra el origen de la referencia.** representa un ejemplo de aprendizaje no supervisado. En este caso, el modelo recibe un conjunto de datos de entrada sin etiquetas, compuesto por diferentes tipos de frutas (manzanas, mangos y sandías). El modelo analiza estos datos y, sin recibir instrucciones explícitas sobre las categorías, identifica patrones en los datos y agrupa las

frutas según sus similitudes visuales o características compartidas. Como resultado, el sistema genera tres grupos distintos: uno con manzanas, otro con mangos y otro con sandías.

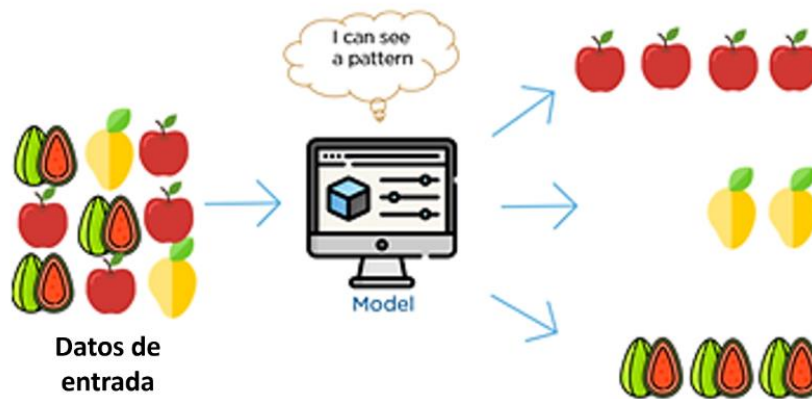


Figura 2-5 Aprendizaje no supervisado [48].

El concepto clave en esta imagen es que el modelo no ha sido entrenado con etiquetas previas, sino que ha aprendido a encontrar estructuras y patrones en los datos por sí mismo. Este tipo de aprendizaje es útil en escenarios como la segmentación de clientes, la detección de anomalías o la agrupación de documentos según su contenido.

Existen varios tipos de algoritmos de aprendizaje no supervisado. Cada uno se ajusta a una necesidad diferente y se pueden englobar en los siguientes tipos [47], [49]:

- **Agrupamiento (*Clustering*):** consiste en organizar datos similares en grupos o clústeres sin que el modelo tenga información previa sobre las categorías. Para ello, los algoritmos calculan distancias entre los puntos de datos y asignan cada uno al grupo más adecuado. Algunos métodos populares son K-Means, DBSCAN y clustering jerárquico. Un ejemplo recurrente es la segmentación de clientes en marketing en función de su comportamiento de compra. El modelo agrupa a clientes con hábitos similares, permitiendo a las empresas personalizar sus estrategias de venta.
- **Asociación:** los modelos de asociación buscan relaciones entre elementos dentro de un conjunto de datos. Funcionan analizando la frecuencia con la que ciertos elementos aparecen juntos y generan reglas basadas en ello. Un algoritmo común en este ámbito es Apriori, que encuentra patrones recurrentes en bases de datos. Por ejemplo, en supermercados, el análisis de cesta de la compra revela que los clientes que compran pan y mantequilla tienen una alta probabilidad de comprar también mermelada. Esto permite a las tiendas optimizar la disposición de los productos.
- **Reducción de dimensionalidad:** se emplea cuando un conjunto de datos tiene muchas variables (o dimensiones) ya que el procesamiento y el análisis pueden volverse ineficientes. Los algoritmos de reducción de dimensionalidad transforman los datos en un conjunto de menor dimensiones manteniendo la mayor cantidad posible de información relevante. Sucede por ejemplo en el reconocimiento facial, un sistema de reducción de dimensionalidad extrae solo las características más representativas, como la forma de los ojos o la distancia entre la nariz y la boca, en lugar de analizar cada píxel de la imagen.
- **Detección de anomalías:** se centra en encontrar datos que se desvían significativamente del comportamiento normal dentro del conjunto de datos. Se basa en medir cómo de diferente es un punto en comparación con los demás. Volviendo al ejemplo del fraude en banca, se pueden usar estos algoritmos si un cliente que habitualmente realiza pagos pequeños de repente transfiere una gran cantidad de dinero al extranjero, el sistema puede marcar esta transacción como sospechosa y solicitar verificación adicional.

2.2.3 Aprendizaje semisupervisado

Como se ha expuesto en los apartados previos, los algoritmos de aprendizaje supervisado emplean datos etiquetados para llevar a cabo su entrenamiento, mientras que los algoritmos de aprendizaje no supervisado trabajan con datos no etiquetados. Las técnicas de aprendizaje semisupervisado combinan ambos tipos de datos en su proceso de aprendizaje. Este enfoque es especialmente útil cuando la obtención de datos etiquetados es costosa o laboriosa, pero se dispone de una gran cantidad de datos sin etiquetar [50].

Son las limitaciones tanto del aprendizaje supervisado como del no supervisado las que indujeron el origen de este concepto. El aprendizaje supervisado ofrece gran precisión, pero tiene un coste computacional elevado, debido al proceso de etiquetado de los datos, que requiere la intervención de expertos en el área de estudio. El aprendizaje no supervisado, no alcanza la precisión del supervisado, pero es una metodología más accesible y su costo computacional y temporal es menor, ya que no depende del etiquetado de los datos. En este contexto, el aprendizaje semisupervisado se presenta como una solución intermedia, combinando la precisión del aprendizaje supervisado con una optimización de los costos computacionales mediante el uso de datos no etiquetados.

Como se entiende de la Figura 2-6, el aprendizaje semisupervisado se basa en la premisa de que, al incorporar datos no etiquetados en el proceso de entrenamiento, es posible mejorar la precisión y generalización del modelo. Esto se debe a que los datos no etiquetados pueden contener información estructural que ayuda a capturar mejor la distribución de los datos [51]. Existen dos principales métodos en aprendizaje semisupervisado: el algoritmo de consistencia y los métodos de ensamble. En el primero se entrenan modelos para que sus predicciones sean consistentes bajo perturbaciones o transformaciones de los datos no etiquetados. En los segundos se utilizan múltiples modelos que generan pseudoetiquetas, que son etiquetas generadas automáticamente por un modelo a partir de datos no etiquetados, en contraste con las etiquetas del aprendizaje supervisado, que son asignadas manualmente por un experto humano [52], para los datos no etiquetados. Las predicciones consensuadas se utilizan para mejorar el entrenamiento del modelo [53].

Aunque el aprendizaje semisupervisado ofrece ventajas significativas, también presenta algunos desafíos. Por ejemplo, la necesidad de datos no etiquetados de calidad, que en caso de contener ruido o estar sesgados, pueden introducir errores en el modelo. Otro desafío es la correcta selección de hiperparámetros, es crucial determinar el equilibrio adecuado entre los datos etiquetados y no etiquetados para el rendimiento del modelo. Y por último llevar a cabo suposiciones incorrectas, ya que si las suposiciones sobre la distribución de los datos (como la suposición de agrupamiento) no se cumplen, el rendimiento del modelo puede verse afectado negativamente.

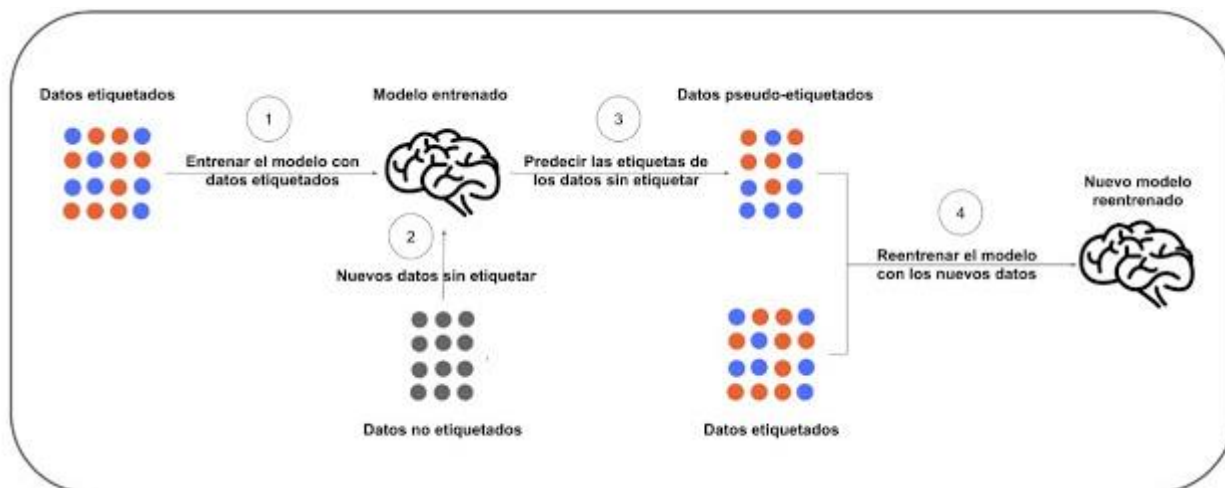


Figura 2-6 Aprendizaje semisupervisado [53].

En la actualidad encontramos mecanismos de aprendizaje semisupervisado en importantes ámbitos como [54]:

- En el procesamiento de imágenes médicas: mientras la clasificación de imágenes requiere la intervención de expertos en salud, el aprendizaje semisupervisado permite aprovechar grandes volúmenes de imágenes sin etiquetar para mejorar la precisión diagnóstica sin la necesidad de una anotación manual exhaustiva.
- En el ámbito del PLN: se emplea en tareas como el análisis de sentimientos o la traducción automática, donde el entrenamiento con etiquetado es un proceso intensivo en recursos. Gracias al aprendizaje semisupervisado, los modelos pueden mejorar su desempeño aprendiendo patrones a partir de datos no etiquetados, reduciendo así la necesidad de intervención humana.
- En el reconocimiento de voz: donde permite entrenar modelos con grabaciones de audio sin transcribir, lo que reduce significativamente el costo asociado a la transcripción manual y mejora la capacidad del sistema para interpretar el habla de manera más precisa.

2.2.4 Aprendizaje por refuerzo

El aprendizaje por refuerzo es una técnica de aprendizaje automático en la que se aprende a tomar decisiones óptimas mediante la interacción, buscando maximizar una recompensa acumulada a lo largo del tiempo. Este enfoque se basa en la idea de que el agente, a través de ensayo y error, descubre qué acciones conducen a resultados positivos o negativos, ajustando su comportamiento en consecuencia [55].

Para comprender bien el funcionamiento de este tipo de aprendizaje hay que explicar algunos términos clave:

- Agente: sistema o entidad que aprende a tomar decisiones. Puede ser un robot, un software o una IA.
- Entorno o ambiente: espacio en el que el agente opera e interactúa. Puede ser un videojuego, una ciudad para un coche autónomo o una bolsa de valores en un sistema financiero.
- Estado: representa la situación actual del entorno con la que el agente interactúa.
- Acción: decisión que toma el agente en un estado determinado.
- Recompensa: valor que el agente recibe después de realizar una acción. Puede ser positiva (cuando la acción es beneficiosa) o negativa (cuando la acción es perjudicial).
- Política: estrategia que sigue el agente para decidir qué acción tomar en cada estado.
- Función de valor: es una estimación de la utilidad a largo plazo de cada estado o acción, lo que permite al agente aprender qué estrategias son más efectivas.

Por ejemplo, imaginemos un coche autónomo que utiliza aprendizaje por refuerzo para mejorar su conducción en una ciudad. El agente es el coche autónomo, que debe aprender a conducir de manera segura y eficiente. Su entorno es la ciudad, con semáforos, peatones y otros vehículos. En cada momento, el coche percibe su estado, que incluye información como la velocidad actual, la distancia a otros coches y si el semáforo está en rojo o verde. Para cada estado, el coche debe elegir una acción, como frenar, acelerar o girar. Después de ejecutar la acción, recibe una recompensa: si respeta el semáforo en rojo, obtiene una recompensa positiva; si pasa en rojo y causa un accidente, recibe una recompensa negativa. A lo largo del tiempo, sigue una política, es decir, una estrategia para tomar decisiones que maximicen su seguridad y eficiencia. Su función de valor le permite evaluar qué acciones generan los mejores resultados en el largo plazo. Con este proceso de prueba y error, el coche mejora su capacidad de conducción y aprende a reaccionar correctamente en diferentes situaciones de tráfico.

Como podemos observar en la Figura 2-7, en el aprendizaje por refuerzo, el agente observa el estado actual del entorno, selecciona una acción según una política (estrategia de decisión), recibe una recompensa del entorno y observa el nuevo estado resultante. El objetivo del agente es aprender una política óptima que maximice la recompensa acumulada. Este proceso se formaliza comúnmente mediante Procesos de Decisión de Markov (MDP), que proporcionan un marco matemático para modelar

la toma de decisiones en situaciones donde los resultados son parcialmente aleatorios y controlados por el agente.

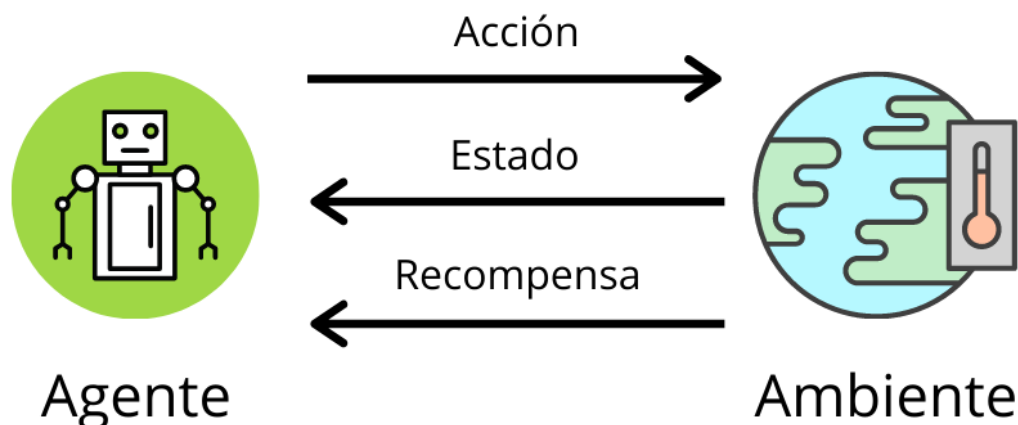


Figura 2-7 Aprendizaje por refuerzo [56].

El aprendizaje por refuerzo ha demostrado ser eficaz en una amplia gama de aplicaciones, desde el control de sistemas autónomos hasta la predicción de fenómenos naturales. Por ejemplo, se ha implementado en modelos de predicción de la precipitación mensual, donde el agente aprende a identificar patrones en datos meteorológicos para mejorar la precisión de las predicciones [57]. También ha cobrado gran importancia gracias a su combinación con técnicas de aprendizaje profundo, que ha dado lugar al *Deep Reinforcement Learning*. Este permite al agente manejar espacios de estado y acción de alta dimensionalidad. Este enfoque ha sido fundamental en el desarrollo de sistemas avanzados, como los agentes que pueden aprender a jugar videojuegos complejos o controlar robots en entornos dinámicos [58].

Es importante destacar que el aprendizaje por refuerzo difiere de otros paradigmas de aprendizaje automático, como el aprendizaje supervisado y no supervisado. Mientras que el aprendizaje supervisado utiliza datos etiquetados para entrenar modelos y el no supervisado busca encontrar estructuras ocultas en datos sin etiquetar, el aprendizaje por refuerzo se centra en la toma de decisiones secuenciales y en cómo las acciones actuales afectan las recompensas futuras. Es decir, permite a los agentes aprender comportamientos óptimos a través de la interacción con su entorno, basándose en la retroalimentación recibida en forma de recompensas o castigos. Su capacidad para abordar problemas complejos de toma de decisiones lo convierte en una herramienta esencial en el campo del aprendizaje automático y la IA.

2.2.5 Deep Learning y las redes neuronales

Como parte del estudio del ML, aparece la subdisciplina llamada aprendizaje profundo o *Deep Learning* (DL). Se caracteriza por la capacidad de utilizar enormes cantidades de datos y del uso de numerosos algoritmos diferenciados en capas para dar al ordenador capacidades de análisis y razonamiento similares a la del ser humano, dotando a la máquina de independencia para aprender y realizar tareas propias, como la identificación de imágenes, el reconocimiento del lenguaje y la predicción de variables.

La principal distinción entre DL y ML radica en la arquitectura de la red de capas utilizada. Mientras que los modelos tradicionales de ML suelen emplear redes con una o dos capas, los modelos de DL incorporan estructuras mucho más profundas, con tres, cientos o incluso miles de capas, lo que les permite extraer patrones y características de los datos de manera más sofisticada y eficiente. Estas capas, están compuestas por la unidad de pensamiento o neurona que se entrelaza en redes formando las denominadas redes neuronales [59].

Las redes neuronales son consideradas modelos de los más capaces que existe en Machine Learning. Funcionan de tal manera que cada neurona asume una función matemática y gracias a la combinación y

coordinación de estas, pasándose sus resultados las unas a las otras, son capaces de crear una red. Se conocen como redes profundas cuando son de gran tamaño y tienen numerosas capas.

A nivel estructural, como se ve en la Figura 2-8, las redes de neuronas están organizadas en distintos tipos de capas, cada una con una función específica en el proceso de aprendizaje: capas de entrada, capas ocultas y capa de salida. La capa de entradas son las encargadas de recibir los datos iniciales del entorno. Cada neurona en esta capa representa una característica específica de los datos de entrada. Las capas ocultas son aquellas que no tienen conexión directa con el exterior y en las que ocurre el procesamiento interno del modelo. Estas capas permiten a la red neuronal extraer patrones, optimizar el aprendizaje y mejorar la precisión de los resultados. Y, por último, la capa de salida es la responsable de generar la respuesta final de la red neuronal, representando la información procesada y transformada a partir de las capas anteriores [60].

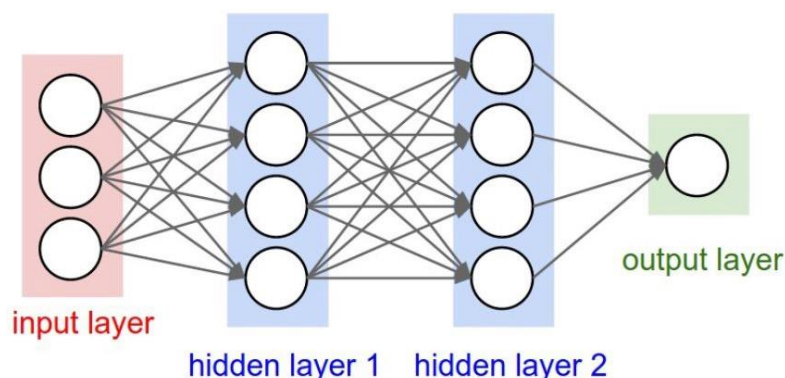


Figura 2-8 Capas de una red de neuronas [61].

Las redes neuronales profundas están inspiradas en la estructura y funcionamiento del cerebro humano. Su aprendizaje se basa en un modelo jerárquico en el que las capas inferiores procesan conceptos básicos y, a medida que se avanza hacia capas superiores, se capturan patrones más complejos. Por ejemplo, en el reconocimiento de imágenes, las primeras capas pueden identificar formas simples como líneas y bordes, mientras que las capas intermedias detectan patrones más elaborados y, finalmente, las capas superiores reconocen objetos completos. Cuantas más capas posee una red neuronal, mayor es su capacidad para extraer información abstracta a partir de los datos de entrada [62]. Por esto, las capas ocultas desempeñan un papel crucial en la capacidad de aprendizaje de la red. Cuantas más capas ocultas posea un modelo, mayor será su capacidad para identificar patrones complejos y producir resultados más precisos. Este enfoque escalonado permite mejorar la precisión y optimización del aprendizaje en modelos de DL [63].

2.3 Desarrollo de los LLMs

Los LLMs son modelos de inteligencia artificial entrenados en enormes volúmenes de texto para comprender y generar lenguaje natural de manera coherente y contextualizada. Estos modelos están basados en redes neuronales profundas y han sido optimizados para diversas tareas del procesamiento del lenguaje natural (PLN, *Natural Language Processing*), como la traducción automática, la generación de texto, el análisis de sentimientos y el resumen de documentos.

2.3.1 Origen y desarrollo

Antes de que existieran los LLMs, los ordenadores usaban métodos más simples para entender y predecir el lenguaje. Dos de estos métodos eran los modelos de Markov y los modelos de n-gramas, que se basaban en contar y analizar las palabras de un texto para hacer predicciones. Estos usaban estadísticas para predecir palabras basándose en su frecuencia en textos previos, pero tenían muchas limitaciones para comprender el significado real de una oración. No llegaban a comprender el contexto que se estaba tratando.

El NLP ha evolucionado significativamente en las últimas décadas gracias a la introducción de modelos capaces de representar palabras y frases en forma de vectores numéricos, permitiendo así una mejor comprensión y procesamiento del texto por parte de las computadoras. A continuación, se presentan algunos hitos clave en esta evolución.

2.3.1.1 Representaciones distribucionales de palabras

En 2013, en [52] se introdujo Word2Vec, un modelo de aprendizaje profundo basado en redes neuronales para representar palabras en un espacio vectorial de alta dimensión. Este se basa en dos métodos principales: el Skip-gram y el CBOW. El primero predice palabras de contexto a partir de una palabra dada. Y el segundo, cuyas siglas significan *Continuous Bag of Words*, predice una palabra objetivo utilizando su contexto. La gran ventaja de Word2Vec es que captura relaciones semánticas entre palabras a través de operaciones algebraicas en sus vectores. Por ejemplo, al realizar la operación "rey - hombre + mujer", el resultado del vector resultante es cercano al vector de la palabra "reina".

Este modelo revolucionó el campo del PLN al demostrar que es posible representar el significado de las palabras de manera distribuida y en un espacio continuo, permitiendo aplicaciones como la detección de sinónimos, la traducción automática y el análisis semántico de textos.

2.3.1.2 Representaciones basadas en coocurrencia global

A pesar del éxito de Word2Vec, una de sus limitaciones era que únicamente capturaba información local basada en ventanas de contexto cercanas, sin aprovechar información de coocurrencia global en el corpus². Para abordar este problema, en 2014 en [53] desarrollaron GloVe (*Global Vectors for Word Representation*).

GloVe combina las ventajas de Word2Vec con modelos basados en matrices de coocurrencia. En lugar de entrenar una red neuronal, GloVe utiliza estadísticas globales de coocurrencia para construir vectores de palabras más representativos. Esto permite que el modelo capte relaciones semánticas de manera más eficiente y logre un mejor rendimiento en tareas como la detección de analogías.

2.3.1.3 Modelos secuenciales

Los primeros enfoques de redes neuronales en PLN se basaban en Redes Neuronales Recurrentes (RNNs), que presentaban problemas al manejar largas secuencias de texto debido a la desaparición del gradiente durante el entrenamiento. Para solucionar este problema, en 1997, en [54] se introducen las LSTM (*Long Short-Term Memory networks*), un tipo de RNN mejorado que permite capturar dependencias a largo plazo en secuencias de texto. Estas, introdujeron células de memoria y puertas de control para gestionar qué información debe ser retenida o descartada en cada paso del procesamiento de la secuencia. Posteriormente, en [55] se propusieron las GRU (*Gated Recurrent Units*), una variante simplificada de las LSTM que ofrece un desempeño similar con una arquitectura más ligera.

Estos modelos se utilizaron ampliamente en tareas de PLN como traducción automática, reconocimiento de voz y análisis de sentimientos. Sin embargo, las LSTM y GRU aún procesaban la información de manera secuencial, lo que limitaba la escalabilidad y el paralelismo en el entrenamiento de modelos más grandes.

2.3.1.4 Transformers

En 2017, en [22] presentaron la arquitectura Transformer, que revolucionó el PLN al eliminar la necesidad de procesar secuencias de texto de manera secuencial. El Transformer se basa en el concepto de autoatención (*self-attention*), que le permite procesar todas las palabras de una secuencia de manera simultánea, en lugar de hacerlo de manera secuencial como las RNNs. Esto mejora significativamente la eficiencia y permite capturar relaciones de largo alcance en el texto. Esto resolvió varios problemas de las redes LSTM y GRU, como el manejo de dependencias a largo plazo y la capacidad de

² En el NLP, un corpus es el conjunto de textos utilizados para entrenar, evaluar o desarrollar modelos de lenguaje.

paralelización en el entrenamiento, que aumentaron considerablemente; o la reducción del problema del gradiente desaparecido, consiguiendo mantener la relevancia de todas las palabras, no como en las RNNs, donde las primeras palabras de una oración tienen menos impacto en la predicción porque la información se diluye con el tiempo.

Gracias a esta arquitectura, se desarrollaron modelos a gran escala, mucho más avanzados, como BERT (2018), GPT-2 (2019), GPT-3 (2020) y modelos recientes como Llama3, Mistral y DeepSeek, que siguen optimizando el procesamiento del lenguaje natural.

2.3.2 Principios fundamentales de los LLMs

Los LLMs son sistemas de inteligencia artificial basados en DL. Estos modelos utilizan millones o incluso billones de parámetros, lo que les permite generar, comprender y manipular el lenguaje natural con una precisión sin precedentes. El funcionamiento de un LLM se basa en varios principios fundamentales, detallados a continuación:

1. Los modelos de lenguaje no procesan el texto como una secuencia continua de caracteres o palabras, sino que lo dividen en unidades llamadas *tokens*. Un *token* es la unidad mínima de significado en la que se descompone el texto antes de ser procesado por un modelo de lenguaje. Puede ser: una palabra completa ("inteligencia"), parte de una palabra ("inteli-", "-gencia"), un solo carácter ("a", "b", "c"), o un símbolo o signo de puntuación ("", ".", ":", "?").

El uso de *tokens* permite que el modelo trabaje con secuencias de texto de manera eficiente, ya que predice el siguiente *token* en la secuencia basándose en el contexto previo. Este enfoque es clave en modelos como GPT-3 y Llama3, que generan texto palabra por palabra analizando las probabilidades de aparición de cada *token*. Gracias a esto, permite manejar el lenguaje de forma flexible y generar texto coherente. Aunque en algunos ámbitos la división en *tokens* puede ser problemática para algunos idiomas donde las palabras no están claramente separadas, como el chino o el japonés.

2. Los LLMs modernos están basados en la arquitectura *Transformer*, que como ya hemos visto permite que el modelo analice todas las palabras de la oración al mismo tiempo, y así no solo ve qué palabras están juntas, sino que aprenden cómo de importantes son entre sí. Con esto, se consigue que los modelos generen texto más coherente y natural, pero conlleva que el entrenamiento requiere grandes volúmenes de datos y una cantidad considerable de recursos computacionales.

3. Los modelos de lenguaje pasan por dos etapas fundamentales de entrenamiento: preentrenamiento masivo y ajuste fino. En el primero, se entrena al modelo con billones de palabras de diferentes usando diversas fuentes como libros, artículos, páginas web, etc. No está especializado en una tarea específica, sino que aprende las reglas generales del lenguaje. Este proceso es muy costoso en términos de recursos computacionales y suele realizarse en centros de datos con potentes GPU y TPU.

En el segundo, el modelo se especializa en tareas específicas como pueden ser la traducción, la generación de código o incluso la asistencia médica. Para ello, se entrena con bases de datos específicas. De tal manera, permite adaptar un modelo general a tareas específicas sin necesidad de entrenarlo desde cero. Hay que tener en cuenta, que, si el ajuste fino no se realiza correctamente, el modelo puede perder conocimientos generales adquiridos en el preentrenamiento.

4. La capacidad de aprender en contexto (*In-Context Learning*) de los LLMs les permite adaptarse a nuevas tareas sin necesidad de ser reentrenados. En lugar de modificar sus parámetros, el modelo aprende a partir de ejemplos proporcionados en la propia entrada del usuario. Este enfoque permite que los LLMs sean muy versátiles y puedan adaptarse a una variedad de tareas con solo recibir instrucciones adecuadas. Esto ofrece la ventaja de permitir utilizar el modelo para nuevas tareas sin modificar su estructura. Aunque a veces, el modelo puede interpretar mal las instrucciones y generar respuestas incorrectas si el contexto no está claro.

2.3.3 El desarrollo de los modelos a gran escala

El año 2018 marcó un punto de inflexión en el campo del PLN con la llegada de los primeros modelos preentrenados a gran escala. Estos modelos, basados en la arquitectura *Transformer*, introdujeron una nueva manera de procesar el lenguaje mediante el preentrenamiento en grandes volúmenes de datos seguido de *fine-tuning* en tareas específicas. Desde entonces, la evolución de los LLMs ha permitido avances significativos en la generación de texto, la comprensión del lenguaje y la interacción hombre-máquina.

La llegada del primer modelo de la familia *Generative Pre-trained Transformer* (GPT), desarrollado por OpenAI fue el mismo año, y presentado en [64]. GPT-1 introdujo el enfoque de preentrenamiento seguido de *fine-tuning*, lo que permitió que el modelo adquiriera un conocimiento general del lenguaje antes de ser ajustado para tareas específicas. Fue preentrenado con un corpus masivo de libros digitales y utilizó aprendizaje autoregresivo, donde el modelo predice la siguiente palabra basándose en el contexto previo. Aunque su rendimiento fue notable, todavía estaba limitado en su capacidad de generar respuestas coherentes en textos largos.

Pocos meses después de la publicación de GPT-1, Google lanzó BERT (*Bidirectional Encoder Representations from Transformers*), un modelo que revolucionó la comprensión del lenguaje al introducir un enfoque bidireccional en el procesamiento del texto [27]. Se empleó un preentrenamiento con predicción de frases y máscara de palabras en lugar de predicción del siguiente *token* y evolucionó notablemente la capaz de comprender el lenguaje, mejorando tareas como: análisis de sentimientos, respuestas a preguntas y clasificación de texto.

En 2019 OpenAI lanzó GPT-2, aumentando significativamente la escala del modelo y demostrando capacidades emergentes en generación de texto [65]. Fue el primer modelo de generación avanzada al contar con 1,5 mil millones de parámetros, un gran salto comparado a los 117 millones de GPT-1. Preentrenado con un corpus de texto extraído de internet, mostró habilidades sorprendentes en la generación de textos coherentes y extensos. El modelo GPT-2 fue el primer LLM de código abierto publicado, permitiendo que otros desarrolladores crearan y entrenaran modelos propios. Sin embargo, OpenAI decidió no liberar el modelo completo en un inicio por preocupaciones sobre el uso indebido de su tecnología. Fue el primero de muchos de una tendencia de publicar modelos en abierto que todavía perdura.

En el mismo año, Google Research presentó T5 (*Text-to-Text Transfer Transformer*), un modelo de código abierto que reformuló todas las tareas de procesamiento del lenguaje natural como problemas de conversión de texto a texto [66]. Creó un enfoque unificado, en lugar de entrenar modelos distintos para diferentes tareas, T5 convirtió todas las tareas en un mismo formato de entrada-salida. Lo preentrenaron con el corpus C4 (*Colossal Clean Crawled Corpus*), una versión filtrada de datos de *Common Crawl*³. Debió la eficiencia en su entrenamiento al uso de una arquitectura de *Transformer encoder-decoder*, a diferencia de GPT, que solo utiliza el decodificador. Y destacó en aplicaciones como traducción automática, resúmenes, generación de texto y respuestas a preguntas. Gracias a este modelo la metodología de texto-a-texto simplificó la adaptación del modelo a distintas tareas y sirvió como base para modelos posteriores.

En paralelo al desarrollo de GPT-2 y de T5, Facebook AI (ahora Meta AI) introdujo BART (*Bidirectional and Auto-Regressive Transformer*), un modelo de código abierto que combinó lo mejor de los enfoques autoregresivos (GPT) y bidireccionales (BERT) [67]. El preentrenamiento con tareas de denoising, es decir, el modelo fue entrenado eliminando palabras aleatorias y obligándolo a reconstruir el texto original, lo que mejoró su capacidad de generar contenido coherente. Combinó un codificador bidireccional (como BERT) y un decodificador autoregresivo (como GPT), consiguiendo una estructura

³ Common Crawl es un gran repositorio de datos web de acceso abierto que recopila y almacena copias de páginas web a gran escala, proporcionando un corpus de texto utilizado para entrenar modelos de lenguaje y desarrollar aplicaciones de inteligencia artificial.

híbrida. Se usó principalmente en generación de texto y resúmenes automáticos. BART se convirtió en la base de modelos avanzados como mBART (Multilingual BART, versión multilingüe) y Pegasus, este último especializado en resúmenes de texto.

En 2020, OpenAI presentó GPT-3, aumentando la escala del modelo a 175 mil millones de parámetros, convirtiéndolo en el modelo más grande de ese momento. Rompe la tendencia de modelos de código abierto que empezaron con GPT-2. Este modelo, mejora capacidades como la comprensión de instrucciones sin necesidad de reentrenamiento o la capacidad de aprendizaje *in-context*, permitiéndole responder a nuevas tareas sin *fine-tuning*. Sin embargo, a pesar de su enorme capacidad, GPT-3 tenía grandes defectos como las alucinaciones (respuestas incorrectas o sin sentido), la falta de conocimiento actualizado debido a su entrenamiento en datos históricos, o costes computacionales extremadamente altos [65].

Ese año, Google y Meta AI desarrollaron versiones multilingües de T5 y BART. El modelo mT5 (*Multilingual T5*) era la extensión de T5 entrenada en 101 idiomas con datos de Wikipedia y Common Crawl [68], que mejoró el rendimiento en traducción y comprensión de textos en múltiples idiomas. Y el modelo mBART que fue entrenado en 25 idiomas con traducción automática y generación de texto multilingüe [69]. Fue aplicado en asistentes virtuales y chatbots multilingües. Estas versiones facilitaron el desarrollo de modelos de lenguaje en múltiples idiomas sin necesidad de entrenar modelos específicos para cada lengua. Ambos modelos serían publicados como código abierto.

En 2022, un consorcio global de más de 1000 investigadores lanzó BLOOM (*BigScience Large Open-science Open-access Multilingual Language Model*), el primer modelo de lenguaje de código abierto entrenado con más de 176 mil millones de parámetros [70]. Es un modelo de código 100% abierto, con código y pesos disponibles para la comunidad científica. Ha recibido entrenamiento en 46 idiomas y 13 lenguajes de programación, con arquitectura basada en Transformers y comparable a GPT-3 en capacidad de generación de texto.

A comienzos de 2023, la empresa Perplexity presenta Perplexity AI v1, que tampoco sería de código abierto. Diseñado para combinar PLN con motores de búsqueda tradicionales, tenía capacidad de consultar internet en tiempo real y proporcionar respuestas actualizadas. Además, aseguraba el uso de fuentes verificables, incluyendo referencias explícitas en sus respuestas. Han mejorado su modelo a con la publicación a finales de 2024 de su modelo Perplexity Pro, que ofrece mejor capacidad de integración con bases de datos externas y fuentes académicas. Ofrece una arquitectura híbrida entre LLMs y modelos de recuperación de información⁴.

Ese mismo año, OpenAI lanzó el modelo GPT-4, con capacidades mejoradas de razonamiento y generación de texto [71]. Este modelo tampoco es publicado de código abierto. Algunas de sus innovaciones incluyeron la capacidad multimodal (capacidad de analizar tanto texto como imágenes), mayor precisión y reducción de sesgos en sus respuestas, y mejor rendimiento en razonamiento complejo, permitiéndole resolver problemas más difíciles que los modelos anteriores.

También en 2023, Meta AI lanzó Llama, con el objetivo de proporcionar modelos optimizados y accesibles para investigadores y desarrolladores. Esta familia de modelos se convirtió en una alternativa clave a modelos de OpenAI, ya que son modelos de código abierto, permitiendo su uso en servidores locales sin restricciones de acceso. Su evolución se resume en:

- Llama 1 (2023): disponible en variantes de 7B, 13B y 65B parámetros, con menor consumo de recursos en comparación con GPT-3.
- Llama 2 (2023): versión mejorada, con entrenamiento en más datos y mejor rendimiento en tareas conversacionales [72]

⁴ Algoritmos que buscan y extraen información de bases de datos o la web en tiempo real, por ejemplo, los motores de búsqueda como Google.

- Llama 3 (2024): última versión con mejoras en eficiencia y menor latencia para dispositivos sin GPU de alta gama.

Al mismo tiempo, la empresa Mistral AI, fundada por antiguos investigadores de Google DeepMind, lanzó Mistral 7B, un modelo de lenguaje de código abierto optimizado para un alto rendimiento con solo 7 mil millones de parámetros, pero comparable a modelos mucho más grandes [73]. Ha demostrado mayor eficiencia en inferencia, lo que permite su uso en hardware menos potente y usa técnicas que optimizan el uso de parámetros sin pérdida de calidad. Además, supera en accesibilidad a modelos como GPT-4. Su arquitectura eficiente ha permitido su integración en aplicaciones móviles y sistemas embebidos⁵.

Respecto a Anthropic AI, tras ser fundada en 2021 por ex-investigadores de OpenAI con el fin de crear una IA con valores humanos y segura para su implementación a gran escala. Lanzan en 2023 su primer modelo, Claude 1, para competir con GPT-4. Entrenado con capacidad mejorada de autocensurar respuestas dañinas y evitar generar contenido peligroso, acuñando el enfoque Constitutional AI [74]. Meses más tarde Claude 2 demostró mejor rendimiento en tareas de comprensión de texto y generación de respuestas complejas ya que fue optimizado en generación de código, respuestas conversacionales y procesamiento de documentos largos. Con él aparece el "*AI-assisted alignment*", por el que la IA ayuda a mejorar su propia alineación ética [75]. En 2024 presentan Claude 3, que usa una arquitectura más eficiente en términos de consumo energético, muestra mayor precisión en tareas de inferencia y menor propensión a alucinaciones. Además, comienza un proceso de integración con sistemas empresariales para aplicaciones en *chatbots* empresariales, asistencia legal y entornos donde la precisión y la seguridad son esenciales. Ninguno de sus modelos ha sido de código abierto.

DeepSeek AI es una de las últimas compañías en desarrollar modelos de lenguaje de código abierto muy eficientes, optimizados para correr en servidores locales y dispositivos con menos recursos [76]. Con la publicación de su LLM en 2024, dieron a conocer su arquitectura *Transformer* mejorada con técnicas de cuantización para reducir el consumo de memoria. Ofrece un enfoque en generación de código y PLN especializado en industrias, y la integración con sistemas de bajo costo sin necesidad de infraestructura en la nube.

2.3.4 Desafíos y limitaciones tecnológicas de los LLMs

La implementación y desarrollo de los LLMs presentan desafíos y limitaciones en distintos aspectos, desde el consumo computacional hasta problemas de alineación ética y generación de información incorrecta:

- Generación de información incorrecta y "alucinaciones": un grave problema de los LLMs es su tendencia a generar información incorrecta o inventada. Se ha demostrado que modelos como GPT-4 y Claude 3 pueden generar datos falsos con confianza, lo que representa un problema en áreas sensibles como medicina, derecho o educación.
- Dificultad en razonamiento y tareas complejas: pese a su continua mejora, los LLMs aún presentan deficiencias en razonamiento lógico y matemático. Modelos como GPT-4 y Claude 3 pueden fallar en problemas de pensamiento crítico y resolución de problemas avanzados [77]. A pesar de mejoras en capacidades multimodales, los LLMs aún no pueden realizar tareas de razonamiento de alto nivel como un experto humano.

⁵ Los sistemas embebidos son sistemas computacionales diseñados para realizar funciones específicas dentro de un dispositivo más grande, integrando hardware y software optimizados para eficiencia, fiabilidad y bajo consumo de recursos.

2.4 Introducción al análisis de encuestas

2.4.1 Definición y propósitos

Las encuestas son instrumentos de recolección de datos utilizados para obtener información sobre opiniones, percepciones o características de un grupo de personas [78]. Su diseño se basa en la formulación de preguntas estructuradas que pueden ser abiertas o cerradas, dependiendo del tipo de análisis que se desee realizar. Desde un punto de vista metodológico, las encuestas forman parte de los estudios descriptivos y analíticos, permitiendo recopilar datos de manera eficiente y sistemática. Estas se utilizan ampliamente en ciencias sociales, educación, mercadotecnia y salud pública para la toma de decisiones basadas en evidencia [79].

El propósito principal de una encuesta es obtener información representativa de una población específica con la menor cantidad posible de sesgo. Para ello, es fundamental un adecuado diseño del cuestionario, la selección de la muestra y la elección del método de administración (presencial, telefónico, en línea o en papel) [80]. En el contexto educativo, por ejemplo, las encuestas permiten evaluar la satisfacción de los estudiantes, el desempeño docente y la efectividad de los programas académicos, proporcionando datos esenciales para la mejora continua [81]. Además, en los últimos años, con el avance de la tecnología y la inteligencia artificial, se ha promovido el uso de herramientas automatizadas para el procesamiento y análisis de encuestas, reduciendo la carga de trabajo en la recopilación y procesamiento de respuestas.

2.4.2 Aplicaciones en distintos ámbitos

Las encuestas se han consolidado como una de las herramientas más utilizadas en la recolección de datos en diversas disciplinas, debido a su capacidad para obtener información estructurada y representativa de una población. Su versatilidad les permite ser aplicadas en ámbitos como la educación, la salud, los negocios y la política, facilitando la toma de decisiones basada en evidencia empírica.

- **Educación:** las encuestas se utilizan para evaluar la satisfacción del alumnado, profesorado, la percepción de la calidad docente y la efectividad y viabilidad de los programas académicos. Diversos estudios han demostrado que los datos obtenidos mediante encuestas permiten mejorar los procesos de enseñanza-aprendizaje y diseñar estrategias pedagógicas más efectivas [81].
- **Salud:** las encuestas de salud pública ayudan en diversas maneras, como monitorizar tendencias en enfermedades crónicas, evaluar el impacto de campañas de prevención y detectar problemas en la accesibilidad a los servicios médicos. También se emplean para recoger datos sobre salud mental, adherencia a tratamientos y calidad en la atención hospitalaria [82].
- **Negocios:** en el sector empresarial, las encuestas desempeñan un papel crucial en la investigación de mercados, permitiendo a las empresas conocer las necesidades y preferencias de los consumidores; y, así, optimizar sus estrategias de ventas, mejorar la segmentación de clientes y evaluar la satisfacción del consumidor [83].
- **Política:** en el ámbito político, las encuestas de opinión pública son fundamentales para conocer la percepción ciudadana sobre candidatos, políticas gubernamentales y problemáticas sociales. Se utilizan en campañas electorales para diseñar estrategias de comunicación y evaluar el impacto de discursos y propuestas [84].
- **Ciencias sociales:** en sociología y otras ciencias, las encuestas permiten analizar fenómenos como la desigualdad social, la percepción de la seguridad y la confianza en las instituciones [85].

La amplia aplicación de las encuestas en distintos campos ha fomentado el desarrollo de metodologías avanzadas para su análisis. En los últimos años, el uso de inteligencia artificial y técnicas de análisis de datos ha optimizado la interpretación de los resultados, permitiendo procesar grandes volúmenes de información de manera más eficiente y precisa [86].

2.4.3 Importancia del análisis automatizado

El análisis de encuestas ha evolucionado significativamente con la digitalización y la implementación de herramientas de procesamiento automatizado. Tradicionalmente, la recopilación y evaluación de encuestas requería un gran esfuerzo manual, lo que aumentaba el tiempo de respuesta y el riesgo de errores humanos en la transcripción e interpretación de los datos. Con los avances en IA y aprendizaje automático, el análisis automatizado de encuestas permite procesar grandes volúmenes de información con mayor rapidez y precisión, facilitando la extracción de ideas clave para la toma de decisiones.

El procesamiento manual de encuestas puede ser un proceso laborioso, especialmente cuando el volumen de respuestas es alto. La automatización permite analizar encuestas en tiempo real, generando reportes instantáneos con gráficos y tendencias que facilitan la toma de decisiones informadas [80]. Este aspecto es particularmente relevante en entornos donde se requiere una respuesta rápida, como en estudios de mercado y evaluaciones de desempeño académico [87].

Además, las técnicas de PLN han permitido la automatización del análisis de respuestas abiertas, lo que antes representaba un desafío debido a la subjetividad del lenguaje. Modelos de IA pueden clasificar sentimientos, extraer palabras clave y agrupar respuestas similares para generar un resumen representativo de los datos. Por ejemplo, en el ámbito educativo, el uso de IA en encuestas de satisfacción permite a las universidades identificar patrones en la percepción estudiantil y mejorar la planificación académica basándose en datos objetivos [81]. En el sector empresarial, el análisis automatizado permite a las empresas ajustar sus estrategias en función de la valoración del cliente sin necesidad de una revisión manual exhaustiva [83].

2.4.4 Métodos estadísticos y el uso de IA en el análisis

El análisis de encuestas requiere herramientas y métodos estadísticos que permitan transformar los datos recopilados en información significativa. Tradicionalmente, se han empleado técnicas estadísticas descriptivas e inferenciales para analizar las respuestas, pero en los últimos años, la IA ha mejorado significativamente la capacidad de extraer patrones y realizar análisis avanzados de los datos.

2.4.4.1 Métodos estadísticos en el análisis de encuestas

Los métodos estadísticos aplicados en el análisis de encuestas se pueden dividir en dos grandes categorías: estadística descriptiva y estadística inferencial. Sus técnicas permiten una comprensión detallada de los datos recopilados y son fundamentales en estudios de mercado, educación y ciencias sociales [80]. La estadística descriptiva se centra en la organización, resumen y presentación de los datos. Algunas técnicas utilizadas incluyen:

- Frecuencias y porcentajes para analizar respuestas categóricas.
- Medidas de tendencia central (media, mediana y moda) para cuantificar respuestas numéricas.
- Medidas de dispersión (desviación estándar, varianza) para evaluar la variabilidad en los datos.

La estadística inferencial se utiliza para hacer generalizaciones sobre una población a partir de una muestra representativa. Algunas de las técnicas más comunes incluyen:

- Pruebas de hipótesis como t de Student o chi-cuadrado para evaluar diferencias entre grupos.
- Análisis de regresión para determinar relaciones entre variables.
- Modelos de clasificación y agrupamiento (*clustering*) para segmentar los datos.

2.4.4.2 Aplicación de la inteligencia artificial en el análisis de encuestas

Con el crecimiento del volumen de datos y la necesidad de realizar análisis más precisos y eficientes, la IA ha ganado protagonismo en el procesamiento de encuestas. Entre las principales aplicaciones de la IA en este campo destacan:

- PLN: se emplea en el análisis de respuestas abiertas para identificar temas recurrentes, sentimientos y opiniones de los encuestados sin necesidad de intervención manual.
- Modelos de aprendizaje automático: como redes neuronales, árboles de decisión y SVM permiten clasificar respuestas, predecir tendencias y detectar patrones ocultos en los datos.
- Automatización de reportes: los sistemas basados en IA pueden generar informes automáticos con resúmenes de los datos y gráficos dinámicos, lo que facilita la interpretación de los resultados sin necesidad de análisis manual [87].

El uso combinado de métodos estadísticos tradicionales e IA ha optimizado el análisis de encuestas, reduciendo el tiempo de procesamiento y aumentando la precisión en la interpretación de los datos. Estos avances han permitido que sectores como la educación, la salud y el marketing implementen estrategias basadas en información más fiable y detallada.

2.4.5 Tipos de encuestas

Las encuestas pueden clasificarse según diversos criterios. Uno de los enfoques más comunes es la clasificación según el medio de administración, diferenciando entre encuestas digitales y encuestas en papel. Ambas tienen ventajas y limitaciones que influyen en su elección dependiendo del contexto de aplicación [80].

2.4.5.1 Encuestas digitales

Las encuestas digitales se realizan a través de medios electrónicos como plataformas web, aplicaciones móviles y correos electrónicos. Su uso ha aumentado considerablemente debido a la accesibilidad y la rapidez en la recopilación y análisis de los datos [82]. Algunas de sus ventajas incluyen:

- Automatización del procesamiento de respuestas: los datos pueden ser analizados en tiempo real con herramientas de análisis estadístico e IA.
- Accesibilidad y alcance global: permiten llegar a una audiencia amplia sin restricciones geográficas.
- Reducción de costos: eliminan los gastos asociados con impresión, distribución y transcripción manual de datos.

No obstante, las encuestas digitales también presentan desafíos, como el sesgo de accesibilidad, ya que algunas poblaciones pueden tener dificultades para acceder a dispositivos electrónicos o internet, lo que podría afectar la representatividad de la muestra. Además, la tasa de respuesta puede ser baja si los encuestados perciben la encuesta como impersonal o poco relevante [83].

2.4.5.2 Encuestas a papel

Las encuestas en papel han sido tradicionalmente el método más utilizado en la recopilación de datos y siguen teniendo relevancia en contextos donde el acceso a tecnología es limitado o donde se requiere un mayor nivel de control en la administración del cuestionario [88]. Algunas de sus características clave son:

- Mayor tasa de respuesta en ciertos contextos: en estudios presenciales, como en aulas de clase o encuestas gubernamentales, los encuestados tienden a responder con mayor frecuencia [89].
- Menor dependencia tecnológica: son útiles en entornos sin acceso confiable a internet o dispositivos electrónicos.
- Mayor percepción de privacidad y confianza: algunas personas prefieren encuestas en papel para compartir información sensible, al no estar vinculadas a direcciones IP o registros electrónicos [90].

A pesar de estas ventajas, las encuestas en papel presentan limitaciones importantes, como el costo de impresión y distribución, la mayor probabilidad de errores en la transcripción manual y la dificultad en el almacenamiento y análisis de datos cuando el volumen de respuestas es alto [81].

En algunos sectores, como el sanitario y el gubernamental, las regulaciones pueden exigir el uso de documentos físicos o respaldos en papel para garantizar la validez de los datos y evitar riesgos asociados con la pérdida de información digital [91]. Por ejemplo, en auditorías médicas y procesos legales, las encuestas en papel pueden servir como evidencia física en caso de disputas o verificaciones oficiales [90]. Por lo que, aunque las encuestas digitales ofrecen ventajas en términos de automatización y análisis de datos, las encuestas en papel siguen siendo esenciales en contextos específicos.

Por todo esto, y gracias al avance de las tecnologías de digitalización y reconocimiento óptico de caracteres, muchas organizaciones han optado por combinar ambos métodos, permitiendo la captura de datos desde formularios en papel y su posterior conversión a formato digital para su análisis automatizado.

2.5 Reconocimiento óptico de caracteres y digitalización

La digitalización de documentos físicos es un proceso esencial para mejorar la accesibilidad, almacenamiento y análisis de la información. En este contexto, el Reconocimiento Óptico de Caracteres o *Optical Character Recognition* (OCR) es una de las tecnologías clave utilizadas para convertir texto impreso o manuscrito en datos digitales editables y analizables.

El OCR ha evolucionado significativamente desde su invención, permitiendo la automatización de tareas que anteriormente requerían una gran cantidad de trabajo manual, como la transcripción de encuestas, formularios y documentos administrativos. En el ámbito del análisis de encuestas en papel, el OCR juega un papel fundamental al facilitar la conversión de respuestas escritas en texto estructurado, lo que permite su posterior procesamiento y análisis [92].

2.5.1 Principios y funcionamiento del OCR

El OCR es una tecnología de procesamiento de imágenes que permite la identificación y conversión de caracteres impresos o manuscritos en texto digital editable. Su funcionamiento se basa en algoritmos de procesamiento de imágenes y reconocimiento de patrones que identifican caracteres individuales y los convierten en texto digital [93]. El proceso de OCR consta de varias etapas clave: preprocesamiento de la imagen, segmentación del texto, reconocimiento y conversión a texto digital, y posprocesamiento y corrección de errores.

1. Preprocesamiento de la imagen: en primer lugar, la imagen del documento debe optimizarse para mejorar la precisión del reconocimiento. Este proceso incluye tratamiento como:
 - Conversión a escala de grises: para reducir el ruido visual eliminando información de color innecesaria.
 - Binarización: dependiendo del gris del tratamiento anterior, convierte la imagen a blanco y negro para resaltar los caracteres y mejorar la segmentación del texto, dejando el fondo perfectamente blanco.
 - Eliminación de ruido: compuesto de técnicas como el filtrado de medianas, para eliminar sombras, y la detección de bordes, que ayudan a mejorar la legibilidad de los caracteres.
 - Corrección de inclinación: que consiste en el ajuste de la orientación del documento para evitar distorsiones en la lectura.
2. Segmentación del texto: en segundo lugar, el OCR segmenta las distintas regiones de interés, diferenciando entre texto, imágenes y otros elementos gráficos. Sucede la segmentación de líneas y caracteres, en la que se identifican bloques de texto, se separan las palabras y caracteres individuales;

y el reconocimiento de estructuras, como encuestas o formularios, donde el OCR debe diferenciar entre casillas de selección, texto de preguntas y respuestas marcadas [94].

3. Reconocimiento y conversión a texto digital: en tercer lugar, el OCR aplica modelos de aprendizaje automático para interpretar los caracteres reconocidos y convertirlos en texto digital. Existen dos enfoques principales: basado en patrones, en el que se compara caracteres con una base de datos de ejemplos predefinidos; y el basado en características, que analiza la estructura de los caracteres (líneas, curvas, intersecciones) para identificarlos [95].
4. Posprocesamiento y corrección de errores: por último, se aplican técnicas de corrección de errores para mejorar la precisión del OCR. Mediante el uso de modelos de lenguaje (tradicionales, no LLM), se utilizan diccionarios y modelos probabilísticos para corregir palabras mal interpretadas; además de realizar una corrección de caracteres que pueden confundirse (por ejemplo, "0" y "O", "1" y "l") [96].

El OCR es una tecnología fundamental en la digitalización de encuestas en papel, ya que permite extraer la información y evitar la transcripción manual. Sin embargo, su precisión puede verse afectada por factores como la calidad del escaneo, el tipo de fuente utilizada y la presencia de escritura manuscrita, lo que requiere herramientas avanzadas de procesamiento para mejorar la tasa de reconocimiento [93].

2.5.2 Aplicación del OCR en este trabajo

El OCR ha revolucionado la forma en que se digitaliza la documentación en papel, permitiendo una digitalización rápida y precisa de los datos. Su implementación en sistemas de análisis de encuestas reduce significativamente el tiempo y los errores asociados a la transcripción manual, optimizando la gestión y el estudio de los resultados obtenidos.

Uno de los principales usos del OCR en encuestas es la automatización de la digitalización y análisis de respuestas. En lugar de insertar manualmente los datos, los sistemas OCR extraen la información automáticamente y la convierten en archivos de texto estructurados, listos para su análisis. Gracias a este proceso, se reduce en gran medida tiempo y costes en comparación con la transcripción manual [92], se minimizan errores humanos al interpretar respuestas escritas a mano o en formularios impresos [80], y aumenta la capacidad de procesar grandes volúmenes de encuestas de forma rápida y eficiente [97]. Por ejemplo, en instituciones educativas, los cuestionarios de evaluación docente que mantuvieron su formato a papel hasta su reciente digitalización a causa de la pandemia de COVID-19. Gracias al reconocimiento de caracteres podría recuperarse el formato en papel ya que pueden ser digitalizados con OCR, permitiendo analizar las respuestas sin intervención manual y generando informes automatizados con los resultados.

El OCR es especialmente útil en encuestas con respuestas estructuradas, como opciones múltiples o escalas de Likert, donde los encuestados deben seleccionar una opción marcada. Para ello detecta qué opción ha sido seleccionada en preguntas de tipo sí/no, opción múltiple o valoraciones numéricas y se combina con técnicas de visión por computadora para diferenciar entre casillas marcadas y sin marcar [98].

Por otro, para las encuestas que cuentan con preguntas abiertas o escritas a mano, el OCR puede interpretar el texto con modelos de reconocimiento de escritura manuscrita, llamados *Intelligent Character Recognition* (ICR) [94]. Estos modelos mejoran la precisión en la conversión de escritura a texto digital y pueden integrarse con técnicas de PLN para análisis semántico [99]. Por ejemplo, en los estudios de mercado, las encuestas con secciones de comentarios abiertos pueden ser procesadas con OCR e interpretadas con IA para identificar tendencias en las opiniones de los clientes [83].

Una vez digitalizadas las encuestas con OCR, los datos pueden integrarse automáticamente en bases de datos o sistemas de gestión para su análisis. De esta manera, los resultados se pueden exportar a formatos como CSV o Excel para procesamiento estadístico y se pueden generar gráficos y reportes automatizados, facilitando la visualización de los resultados [100]. Además, permite la aplicación de modelos de IA y ML para identificar patrones en los datos recolectados [101].

El OCR ha transformado el procesamiento de encuestas en papel, reduciendo la carga de trabajo manual y aumentando la precisión en la recopilación de datos. Y, pese a que pueden cometer errores al interpretar caracteres similares, especialmente cuando la encuesta está escrita a mano o el escaneo tiene ruido visual [87], su aplicación en encuestas estructuradas y abiertas permite digitalizar la información rápidamente, integrándola en bases de datos para su análisis automatizado. Con el avance del ICR y el PLN, el OCR continúa evolucionando para mejorar la interpretación de escritura manuscrita y optimizar el análisis de datos en encuestas.

2.5.3 Modelos de OCR actuales

Existen diferentes modelos OCR, que pueden clasificarse en herramientas de código abierto y soluciones comerciales de pago. Las soluciones de código abierto son utilizadas en múltiples aplicaciones debido a su accesibilidad y flexibilidad, mientras que las herramientas comerciales suelen ofrecer algoritmos más avanzados con mayor precisión y soporte técnico.

2.5.3.1 Comparativa entre OCR de código abierto y OCR de pago

Las herramientas OCR de código abierto y las soluciones comerciales presentan diferencias significativas en diferentes aspectos. Los OCR de código abierto como las librerías TesseractOCR⁶ y Kraken OCR⁷ tienen la ventaja de ser gratuitos y accesibles para cualquier usuario, lo que los convierte en la opción en proyectos de bajo presupuesto. Sin embargo, los OCR comerciales, como ABBYY FineReader⁸ y Google Cloud Vision⁹, requieren una suscripción o licencia de pago, lo que puede ser una barrera para usuarios individuales o pequeñas empresas, aunque a cambio ofrecen mayor precisión y soporte técnico.

	OCR de Código Abierto	OCR de Pago
Precisión en texto impreso	Alta, pero depende de la calidad de la imagen y entrenamiento	Muy alta, con modelos optimizados para distintas fuentes y formatos
Reconocimiento de escritura manuscrita	Limitado en la mayoría de los casos	Avanzado, con ICR entrenados en grandes conjuntos de datos
Flexibilidad y personalización	Muy alta (se puede ajustar y entrenar modelos)	Limitada a las opciones proporcionadas por el software
Requisitos computacionales	Bajo, adaptable a distintos entornos	Puede requerir mayor capacidad de cómputo
Integración con IA/PLN	Necesita herramientas adicionales	Muchas integran ICR
Privacidad y uso local	No depende de servidores externos (procesamiento en local)	Algunas herramientas requieren conexión a la nube
Compatibilidad	Multiplataforma (Linux, Windows, macOS)	Depende del proveedor y puede tener restricciones

Tabla 2-1 Comparación algoritmos OCR (elaboración propia)

En cuanto a la facilidad de uso, las herramientas de código abierto suelen requerir conocimientos técnicos para su instalación y configuración. Por ejemplo, TesseractOCR funciona principalmente a través de línea de comandos y necesita ajustes manuales para optimizar su rendimiento. En cambio, los OCR comerciales están diseñados para ser intuitivos, con interfaces gráficas amigables y flujos de

⁶ Tesseract OCR disponible en <https://github.com/tesseract-ocr/tesseract>

⁷ Kraken OCR disponible en <https://kraken.re/main/index.html>

⁸ ABBYY FineReader disponible en <https://pdf.abbyy.com/es/>

⁹ Google Cloud Vision disponible en <https://cloud.google.com/vision?hl=es>

trabajo optimizados que permiten a los usuarios obtener resultados sin necesidad de conocimientos avanzados en procesamiento de imágenes o IA.

Al estudiar las diferencias técnicas, analizando la Tabla 2-1, se puede observar cómo los OCR de código abierto ofrecen flexibilidad, personalización y procesamiento en local sin depender de servidores externos, pero suelen requerir ajustes manuales y tienen limitaciones en el reconocimiento de escritura manuscrita. Por otro lado, los OCR comerciales destacan por su mayor precisión, integración con IA avanzada y facilidad de uso, aunque pueden requerir más recursos computacionales y, en algunos casos, conexión a la nube. La elección entre ambos depende de la necesidad de personalización, el presupuesto y la prioridad en precisión y automatización.

Por lo tanto, los algoritmos de código abierto son ideales para proyectos donde la flexibilidad y el costo son factores clave, ya que los comerciales, aunque ofrecen mayor precisión y facilidad de uso, requieren una inversión económica.

2.5.3.2 TesseractOCR como solución de código abierto

TesseractOCR es uno de los motores OCR de código abierto más utilizados en la actualidad. Fue desarrollado originalmente por Hewlett-Packard en la década de 1980 y, en 2005, fue liberado como software de código abierto bajo el patrocinio de Google [102]. Desde entonces, ha sido mejorado con soporte para múltiples idiomas, integración con técnicas de preprocesamiento de imágenes y mejoras en el reconocimiento de texto. Las características clave de TesseractOCR son:

- Código abierto y gratuito: permite su uso sin restricciones comerciales.
- Compatibilidad con múltiples idiomas: trabaja en más de 100 idiomas, con modelos entrenados para cada uno.
- Soporte para texto impreso: su rendimiento es óptimo en textos impresos.
- Procesamiento en local: no requiere conexión a internet, garantizando la privacidad de los datos.
- Opciones avanzadas de configuración: permitiendo ajustar parámetros como el modo de segmentación y el umbral de reconocimiento para mejorar la precisión.

Su rendimiento en escritura manuscrita es deficiente y es sensible a la calidad de la imagen, por lo que documentos con baja resolución, ruido o sombras pueden generar errores en el reconocimiento. También tiene dificultades con textos en múltiples columnas o formatos complejos, ya que su segmentación automática no siempre es precisa. Para mejorar su precisión, requiere configuración manual, que puede ser complicada para usuarios sin experiencia. Y su procesamiento es más lento que el de OCR comerciales, ya que no aprovecha de base la aceleración por GPU. A pesar de estas limitaciones, sigue siendo una opción viable en proyectos donde se requiere una solución gratuita y personalizable sin depender de servicios en la nube. Por ejemplo, en el caso de la digitalización de encuestas, TesseractOCR es una opción sólida debido a su integración con sistemas personalizados y su capacidad de procesamiento en local.

3 DESARROLLO DEL TRABAJO

Una vez planteados los objetivos del proyecto, y habiendo estudiado el estado del arte concerniente al ámbito de trabajo, se plantea la necesidad de establecer los requisitos a cumplir con el desarrollo y de elegir cómo proceder con dichas pautas, seleccionando aspectos importantes desde el lenguaje conductor de todos los procesos hasta las herramientas que se quiere emplear con el fin de lograr los mejores resultados en cada proceso y subproceso, y así obtener el mejor producto final.

En este capítulo se realiza un estudio conciso de las elecciones y pasos tomados para llevar a cabo el proyecto hasta conseguir los objetivos. En primer lugar, se presenta el entorno de trabajo escogido, en el que se hablará tanto de GoogleColab como del servidor que el CUD-ENM ha puesto a mi disposición para llevar a cabo todo el desarrollo. Después se presentará el lenguaje escogido como conductor de todos los procesos, que ha sido Python, por su versatilidad y nivel de desarrollo. A continuación, se explican todas las herramientas y bibliotecas, como pueden ser TesseractOCR o ReportLab, que se usan en el proyecto para su funcionamiento. Y para terminar el primer apartado, se presentarán los modelos de LLM seleccionados para llevar a cabo las tareas correspondientes.

En el segundo apartado, se procederá a la explicación de la ejecución y funcionamiento del proyecto. Se dará una introducción a la arquitectura del sistema creado, que servirá para proceder a una explicación clara y organizada de cada uno de los subprocesos involucrados en el proceso principal, de tal manera que se irá explicando cómo se ha planteado y ejecutado cada subproceso, exponiendo los requisitos de cada uno de ellos con el fin de cumplir sus responsabilidades como han sido plantados.

3.1 Entorno de trabajo y herramientas empleadas

Como se ha mencionado en la introducción del capítulo, en este punto se presentan los entornos de trabajo seleccionados: el lenguaje Python como motor del proyecto y las herramientas necesarias para llevar a cabo todos los subprocesos involucrados y los modelos LLM empleados para cumplir con los cometidos establecidos en los objetivos. Para ello se partió de un esquema inicial sobre el que se estudiaron las posibles necesidades que irían surgiendo en cada momento, de tal forma que la elección desde el primer momento no hiciera que malos resultados en los primeros pasos repercutieran negativamente en la eficiencia de los siguientes.

3.1.1 Entornos de trabajo empleados

Estudiando los requisitos del proyecto, lo primero que llama la atención es el empleo de modelos LLM instalados en equipos locales, por lo que la disponibilidad de unidades GPU se hace requisito indispensable, y al no disponer de ellas se recurrió a los siguientes entornos:

3.1.1.1 Entorno de desarrollo GoogleColab

El entorno de desarrollo GoogleColab¹⁰ (como abreviatura de *Google Colaboratory*), visible en la Figura 3-1, es una plataforma online de Google basada en su nube, Google Drive, que permite escribir, ejecutar y compartir código en Python desde el navegador, sin necesidad de instalación local. Está basado en Jupyter Notebook¹¹, que es un famoso entorno de desarrollo colaborativo, usando el mismo formato y ofreciendo una ejecución interactiva y visualización de resultados en la misma celda. Es importante aclarar que el hecho de que sea una herramienta online no implica que los LLMs no estén descargados físicamente en los servidores de Google. Es decir, la finalidad de emplear GoogleColab es utilizar los recursos físicos de Google que los equipos disponibles por el desarrollador son capaces de satisfacer, pero trabaja de la misma manera que un entorno local instalando y ejecutando tanto el código como los modelos LLM.

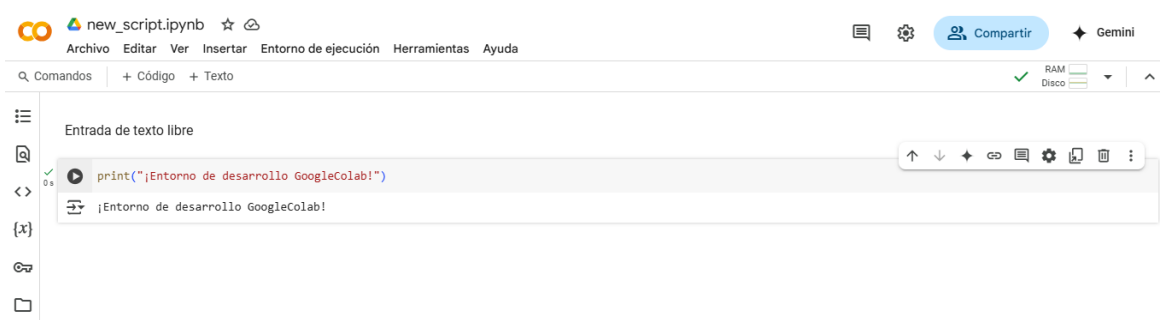


Figura 3-1 Interfaz de GoogleColab (elaboración propia).

Las particularidades que ofrece el empleo de este entorno de desarrollo son varias. En primer lugar, la capacidad de escribir código por secciones, dando la capacidad al desarrollador de ver poco a poco cómo se ejecuta su código y qué resultados genera, haciendo muy efectiva la detección de errores; además, permite intercalar cuadros de texto libre para ayudar a la estructura y entendimiento del código por los demás colaboradores de cada proyecto. En segundo lugar, ofrece la capacidad ya mencionada de trabajo conjunto entre varios desarrolladores que son capaces de trabajar en el mismo script al mismo tiempo.

Por último, la capacidad más significativa que ofrece este entorno de desarrollo, y motivo de selección en este proyecto, es la capacidad que ofrece de ejecución del código con unidades de procesamiento gráfico (GPU) dedicados, ya que son esenciales para la ejecución de los LLM. Existe la capacidad de emplear los LLM con unidades de procesamiento centrales o convencionales (CPU) pero los desarrolladores no lo aconsejan ya que pierde su eficiencia en un porcentaje demasiado importante como para hacer factible la ejecución del proyecto. Como se ve en la Figura 3-2, GoogleColab ofrece diferentes posibilidades de entorno de ejecución, siendo necesaria para nuestro proyecto seleccionar el modelo 'GPU T4' debido al empleo de los LLM.

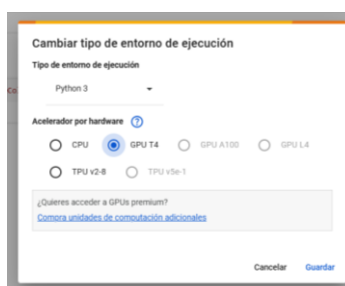


Figura 3-2 Menú de selección de entornos de ejecución (elaboración propia).

¹⁰ GoogleColab está disponible en <https://colab.google/>

¹¹ Jupyter Notebook está disponible en <https://jupyter.org/>

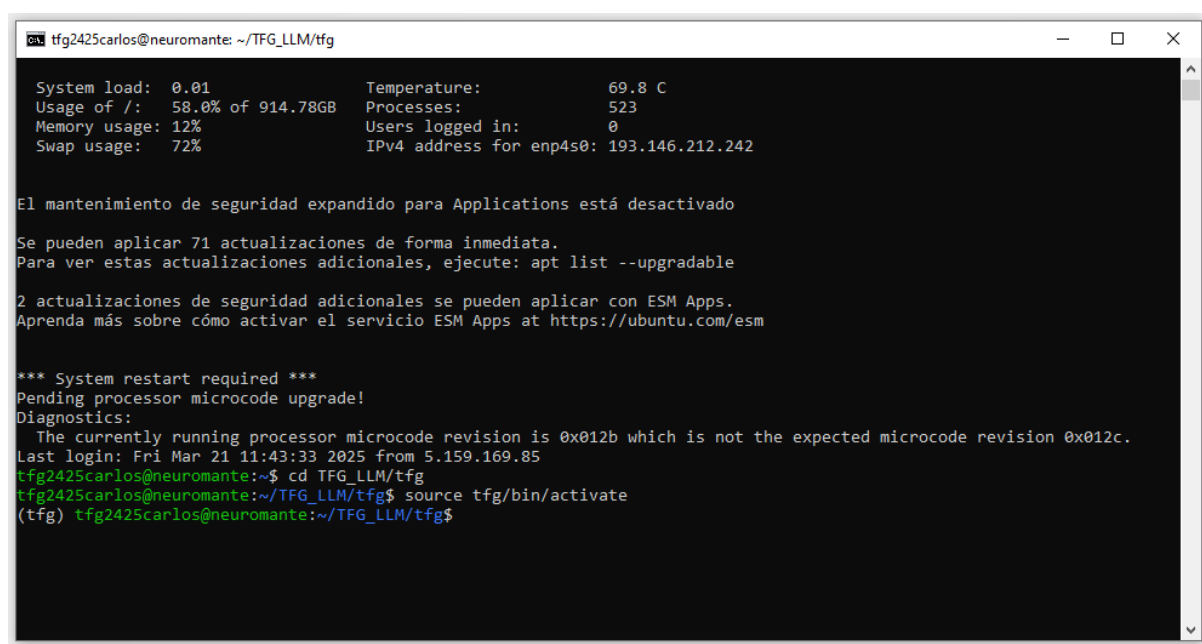
Además del modelo seleccionado, en la Figura 3-2 existen otros dos modelos GPU que no son seleccionables. Se trata de GPUs con mayores capacidades de procesamiento que son accesibles con el plan premium de GoogleColab. Por ello, no han sido aplicables al proyecto actual al no ser de acceso libre como queda establecido en los objetivos.

Cabe recalcar de este entorno de desarrollo que el empleo de forma gratuita de la GPU es limitado, por lo que, una vez superados los límites del plan gratuito dejará de funcionar durante un día, suponiendo una desventaja muy considerable teniendo en cuenta que el número de iteraciones con los LLM son muy numerosas durante el proyecto. Además, el entorno finalizará automáticamente cualquier proceso que lleve veinticuatro horas realizándose, suponiendo un obstáculo en el caso de análisis de larga duración debido a la cantidad de datos a procesar.

3.1.1.2 Servidor empleado

Como ya se ha mencionado en el punto anterior, el proyecto depende de unos recursos considerables a nivel gráfico. Por lo tanto, se ha utilizado un servidor del CUD-ENM para el desarrollo y ejecución del presente proyecto. Este servidor ofrece grandes capacidades de equipo y permite disponer de un entorno que está en funcionamiento en todo momento, sin interrumpir posibles procesos que duren más de un día y que no limita el uso de sus recursos gráficos. Respecto a estos recursos, cuenta con una tarjeta gráfica NVIDIA GeForce RTX 4090, que es el modelo más avanzado de GPU en el mercado para uso de IA en local, creación de contenido gráfico, *gaming*¹² de gran rendimiento y en ciencia de datos y simulaciones.

Este servidor dispone de un sistema operativo de Linux, sobre el que se permite la instalación de software tal como el lenguaje de programación Python y de todas las bibliotecas necesarias en este trabajo, todo ello realizado desde el terminal, como se ve en la Figura 3-3.



```
tfg2425carlos@neuromante: ~/TFG_LLM/tfg
System load: 0.01      Temperature: 69.8 C
Usage of /: 58.0% of 914.78GB    Processes: 523
Memory usage: 12%      Users logged in: 0
Swap usage: 72%        IPv4 address for enp4s0: 193.146.212.242

El mantenimiento de seguridad expandido para Applications está desactivado
Se pueden aplicar 71 actualizaciones de forma inmediata.
Para ver estas actualizaciones adicionales, ejecute: apt list --upgradable

2 actualizaciones de seguridad adicionales se pueden aplicar con ESM Apps.
Aprenda más sobre cómo activar el servicio ESM Apps at https://ubuntu.com/esm

*** System restart required ***
Pending processor microcode upgrade!
Diagnostics:
  The currently running processor microcode revision is 0x012b which is not the expected microcode revision 0x012c.
Last login: Fri Mar 21 11:43:33 2025 from 5.159.169.85
tfg2425carlos@neuromante:~$ cd TFG_LLM/tfg
tfg2425carlos@neuromante:~/TFG_LLM/tfg$ source tfg/bin/activate
(tfg) tfg2425carlos@neuromante:~/TFG_LLM/tfg$
```

Figura 3-3 Interfaz de servidor de CUD mediante el terminal de Windows (elaboración propia).

Sin embargo, el uso del servidor dificulta el desarrollo de código al no disponer de interfaz de usuario, por lo que en paralelo el código se ha desarrollado localmente utilizando la herramienta PyCharm¹³ de JetBrains. Esta herramienta ofrece una interfaz específicamente creada para el desarrollo de software, como se ve en la Figura 3-4, proporcionando ayudas visuales y avisos automáticos durante

¹² Término empleado para referirse a la ejecución de juegos de ordenador, mayoritariamente muy realistas y exigentes de recursos gráficos.

¹³ Herramienta disponible en <https://www.jetbrains.com/es-es/pycharm/>

el desarrollo para auxiliar al desarrollador. Esta herramienta deja crear proyectos que permiten la ejecución de funciones y llamadas de subprocessos de otros scripts.

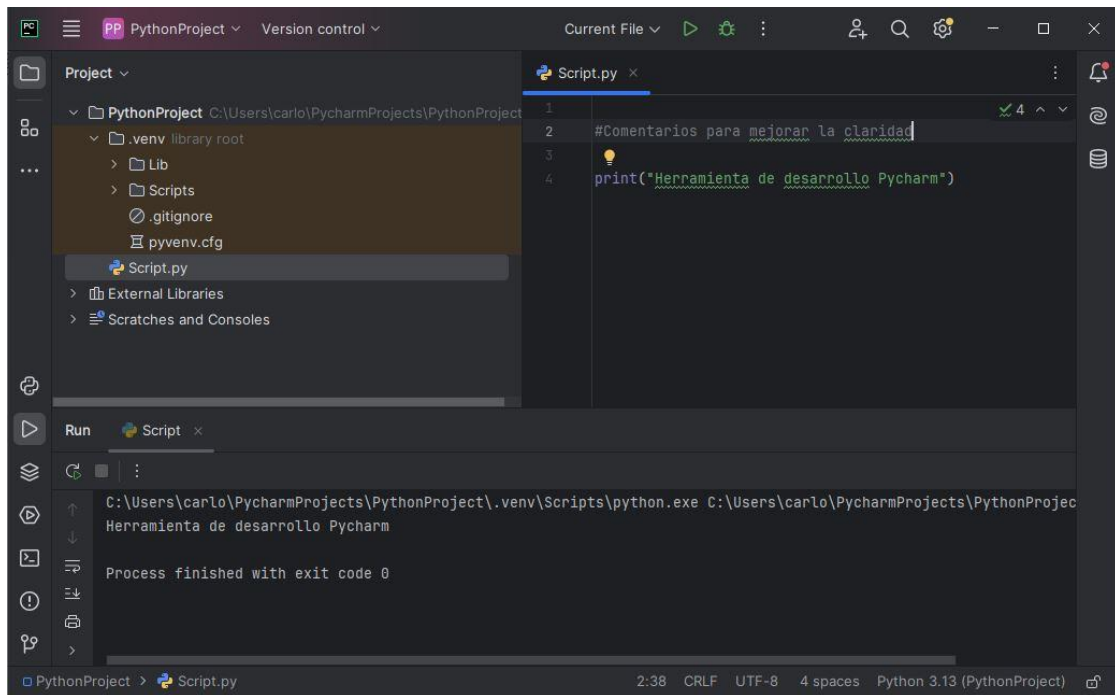


Figura 3-4 Interfaz de desarrollo de PyCharm (elaboración propia).

Para el empleo de este servidor, se requiere de una herramienta externa que permita la descarga y carga de archivos del servidor a nuestro equipo. Para ello, se ha empleado la herramienta FileZilla¹⁴, que permite trabajar de manera intuitiva, como se ve en la Figura 3-5, permitiendo arrastrar archivos del equipo local (visible a la izquierda de la figura) y el usuario del servidor (visible a la derecha de la figura).

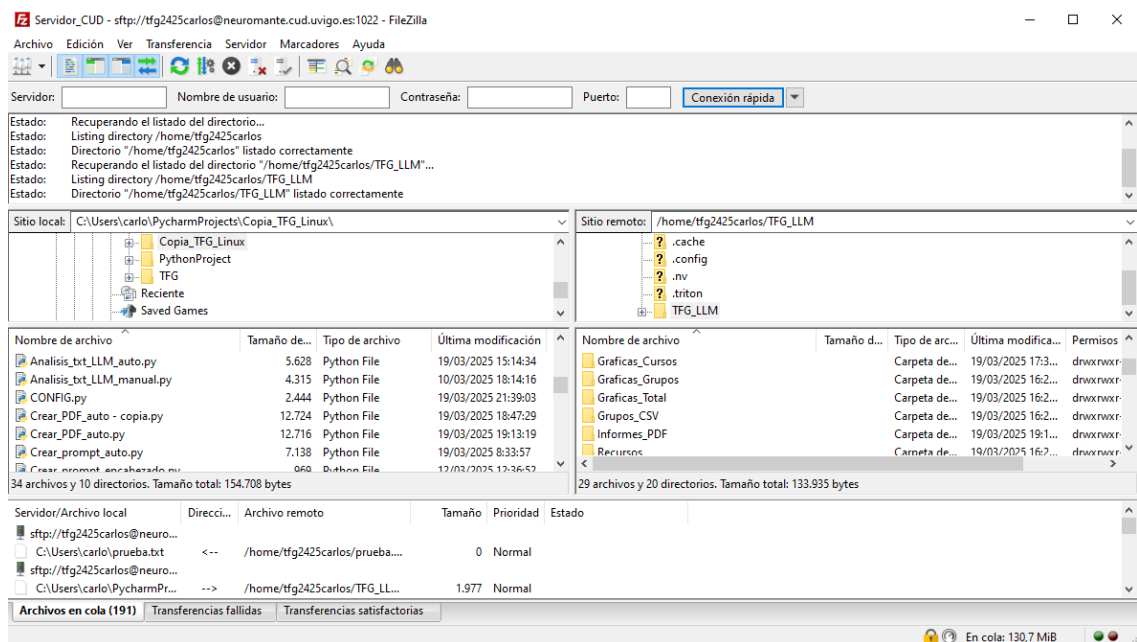


Figura 3-5 Interfaz de FileZilla (elaboración propia).

¹⁴ Herramienta disponible en <https://filezilla-project.org/>

Cuenta el servidor además con un sistema de seguridad que obliga a meter la contraseña y número de seis dígitos generado automáticamente a través de Google Authenticator¹⁵. Este método aporta seguridad al proyecto que sólo puede ser utilizado por los conocedores de la contraseña y que disponen de acceso a la clave de Authenticator. Sin embargo, también supone un retraso importante a la hora de mover un gran número de archivos o a la hora de programar, ya que cada cambio del script supone introducir todas las claves de seguridad.

3.1.2 Lenguaje Python

El lenguaje de programación Python es actualmente el lenguaje de programación más extendido en el campo de la IA, la ciencia de datos y la automatización de procesos. La claridad de su sintaxis permite centrarse en el diseño del sistema, sin necesidad de lidiar con complejidades del lenguaje, facilitando la depuración y mantenimiento del código y permite ejecutar todo el proyecto en local, lo que es crucial en tu caso para preservar la privacidad de los datos recogidos en las encuestas. Para un proyecto que abarca desde el procesamiento de imágenes escaneadas, el análisis con modelos de lenguaje, generación de gráficas y creación de informes en PDF, este lenguaje ofrece una gran versatilidad y una comunidad científica muy activa, que facilita la integración de herramientas avanzadas sin necesidad de desarrollos desde cero.

Python cuenta con bibliotecas específicas para cada uno de los subprocesos del proyecto, ofreciendo bibliotecas de OCR, bibliotecas para manejo de datos, bibliotecas para la generación de gráficas, y bibliotecas para la generación de PDF. Además, Python es el lenguaje estándar en el desarrollo e implementación de LLM gracias al soporte directo de entornos de desarrollo como Hugging Face¹⁶, mediante bibliotecas como *transformers*, que se explica en el siguiente apartado.

Cabe considerar también que cuenta con limitaciones. Por ejemplo, es un lenguaje que no es tan rápido como C++ o Java, puede ser un problema en sistemas con recursos limitados y puede mostrar problemas de paralelización de procesos. Sin embargo, el hecho de que el proyecto esté planteado como una secuencia de módulos que emplea directamente bibliotecas optimizadas, hace que ninguna de las limitaciones expuestas impida el empleo de este lenguaje de programación.

Por ello, teniendo en cuenta todo lo planteado, se elige Python como lenguaje motor del proyecto por su capacidad de desarrollar una solución completa, modular, ejecutable en local, y totalmente automatizada, cumpliendo con los objetivos propuestos de la manera más eficiente. La versión de este lenguaje empleada en el proyecto es Python 3.12.

3.1.3 Librería de python empleadas en el proyecto

Como se ha mencionado en la introducción del capítulo, en este apartado se enumeran y explican todas las bibliotecas necesarias para la ejecución del proyecto. En primer lugar, se expondrán las bibliotecas más genéricas, que son más recurrentes en el uso de Python y posteriormente las bibliotecas específicas que realizan una tarea específica, que podemos llamar herramientas. Las bibliotecas más comunes empleadas son las siguientes:

- Biblioteca *pip*: es la herramienta oficial para descargar bibliotecas desde el repositorio PyPI e integrarlas fácilmente en cualquier proyecto. También facilita la actualización y desinstalación de paquetes según las necesidades del entorno.
- Biblioteca *sys*: proporciona acceso a variables y funciones del entorno de ejecución de Python. Permite interactuar con argumentos pasados al script, finalizar programas o manipular la salida estándar.

¹⁵ Google Authenticator es un software basado en autenticación con contraseña de un solo uso desarrollado por Google.

¹⁶ Hugging Face es una plataforma y empresa que ofrece herramientas y modelos *open-source* de inteligencia artificial, disponible en <https://huggingface.co/>

- Biblioteca *subprocess*: permite ejecutar comandos del sistema operativo desde un programa en Python, ofrece control sobre la entrada, salida y errores de los procesos externos y es útil para automatizar tareas, ejecutar scripts o integrar programas externos en un flujo Python.
- Biblioteca *os*: permite interactuar con el sistema operativo desde Python. Facilita tareas como gestionar archivos, directorios, rutas y variables de entorno, por lo que es fundamental para crear scripts que necesiten acceso al sistema de archivos.
- Biblioteca *re*: permite trabajar con expresiones regulares en Python para buscar, analizar o modificar cadenas de texto. Es útil para identificar patrones complejos dentro de textos, como correos, fechas o formatos específicos.

A continuación, se exponen las bibliotecas específicas para la secuencia del proyecto. Como se mencionó con anterioridad, cada una se expone en este trabajo como una herramienta necesaria para el desarrollo de este:

- Biblioteca *pdf2image*: permite convertir archivos PDF en imágenes, página por página, directamente desde Python, permitiendo la integración de PDFs en flujos de trabajo que requieren tratamiento de imagen, como clasificación o extracción de datos.
- Biblioteca *pytesseract*: actúa como un puente entre Python y el motor OCR de Tesseract, permitiendo extraer texto de imágenes. Es útil para digitalizar documentos escaneados, interpretar formularios o automatizar la lectura de texto visual.
- Biblioteca *pandas*: permite manipular, analizar y estructurar datos de forma eficiente mediante estructuras como *DataFrame*. Es muy útil para leer archivos CSV, realizar filtrados, agrupaciones, operaciones estadísticas y limpieza de datos. Por ello destaca su uso en ciencia de datos y en la automatización de procesos con grandes volúmenes de información.
- Biblioteca *matplotlib*: se utiliza para crear gráficos estáticos, interactivos y personalizados en Python. Permite representar datos mediante diagramas de barras, líneas, histogramas, gráficos de dispersión, entre otros. En este proyecto se emplea módulo *matplotlib.pyplot* específico de esta biblioteca, que proporciona una interfaz de estilo similar a MATLAB para crear gráficos de forma más sencilla.
- Biblioteca *ReportLab*: permite generar archivos PDF de forma programada y personalizada desde Python. Ofrece control total sobre el diseño del documento, incluyendo textos, imágenes, tablas, estilos y posiciones. Suele usarse para automatizar la creación de facturas, certificados, informes o cualquier documento estructurado.

Respecto a las bibliotecas necesarias para el trabajo con LLM encontramos:

- Biblioteca *torch*: se utiliza para construir y entrenar modelos de aprendizaje profundo en Python. Permite trabajar con tensores, realizar operaciones matemáticas aceleradas por GPU y definir redes neuronales de forma flexible.
- Biblioteca *huggingface_hub*: permite interactuar directamente con el repositorio de modelos y datasets de Hugging Face desde Python. Facilita la descarga y gestión de modelos y archivos en el Hub, tanto de forma pública como privada. Controla las claves de acceso API para los modelos en los que sea necesario.
- Biblioteca *transformers*: es una biblioteca de Hugging Face que permite utilizar modelos de lenguaje basados en la arquitectura transformer de forma sencilla en Python. Facilita tareas como generación de texto, resumen, clasificación o traducción con modelos preentrenados. Es clave para integrar inteligencia artificial avanzada en proyectos sin necesidad de entrenar modelos desde cero.

3.1.4 Modelos LLM seleccionados

Como se expone en los objetivos, el desarrollo del proyecto incluye la necesidad de recurrir a modelos LLM para que lleven a cabo diferentes acciones que tradicionalmente llevarían a cabo humanos, con el fin de automatizar por completo el proceso de análisis de encuestas. Sin embargo, queda justificado que los LLMs que se van a emplear son únicamente los de código abierto, disponibles en

repositorios como Hugging Face. Por ello, y habiendo estudiado las opciones actuales, los LLMs seleccionados para estudiar su integración en este proyecto son dos: Llama 3 de Meta, y el modelo R1 de DeepSeek.

Es importante tener en cuenta que, a diferencia de los LLMs avanzados disponibles en internet como los de ChatGPT o Anthropic Claude, los modelos de código abierto seleccionados no son capaces de trabajar con ficheros de ningún tipo, es decir, solamente son modelos de reconocimiento y creación de texto. Por lo tanto, en este proyecto, se tiene que adaptar cada proceso para que el LLM reciba únicamente texto y ajustar los siguientes pasos entendiendo que la respuesta será enteramente texto.

3.1.4.1 Modelo Llama 3 de Meta

Del modelo Llama 3, se ha instalado la versión entrenada con ocho mil millones de tokens, como se puede ver en la Figura 3-6. Ha sido seleccionada porque ofrece un gran equilibrio entre rendimiento, precisión y capacidad de ejecución en entornos locales. Su arquitectura está optimizada para tareas de generación y comprensión de lenguaje natural, lo que lo hace un modelo muy útil para interpretar respuestas de encuestas, redactar conclusiones y generar informes en lenguaje claro y coherente. Y como ya se ha mencionado, se integra fácilmente en Python y permite ejecutar interacciones sin conexión a internet, cumpliendo con el requisito de autonomía establecido para este proyecto.

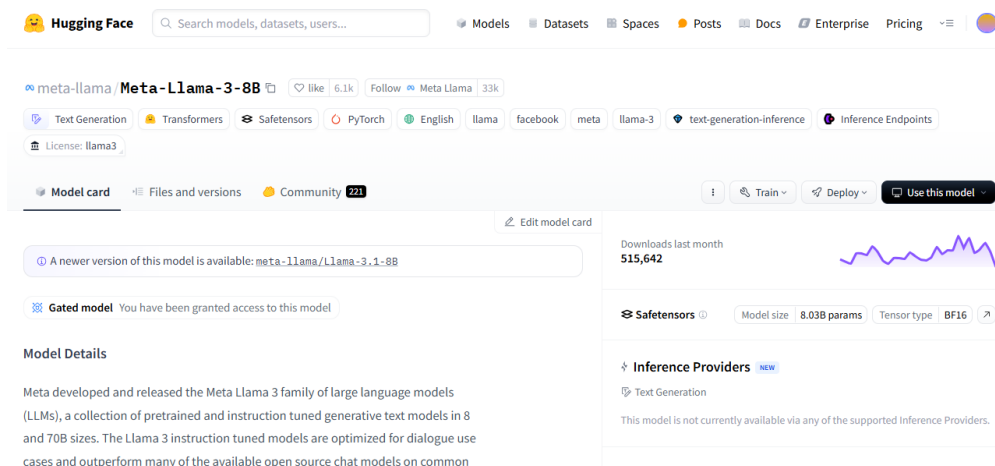


Figura 3-6 Repositorio en Hugging Face de la versión de Llama 3 usada (elaboración propia).

Este modelo ha sido seleccionado por sobresalir entre los LLM de código abierto de occidente, con capacidades iguales o superiores a ChatGPT 3.5 en su versión de setenta mil millones de *tokens* de entrenamiento [103].

3.1.4.2 Modelo R1 de DeepSeek

Del modelo DeepSeek-R1, como se ve en la Figura 3-7, se ha elegido concretamente la versión entrenada con siete mil millones de *tokens*. Ha sido seleccionado porque está diseñado específicamente para tareas de razonamiento, comprensión contextual y análisis estructurado. Esto lo hace especialmente útil para interpretar los datos procesados de las encuestas y redactar conclusiones con alto nivel de coherencia. Este modelo destaca por su capacidad de generar textos explicativos precisos a partir de datos en tablas o entradas semiestructuradas, lo que lo convierte en una herramienta muy eficaz para la fase de análisis y redacción automática del informe. Además, al tratarse de un modelo de código abierto optimizado para uso en local, cumple con los criterios de privacidad y autonomía tecnológica del proyecto.

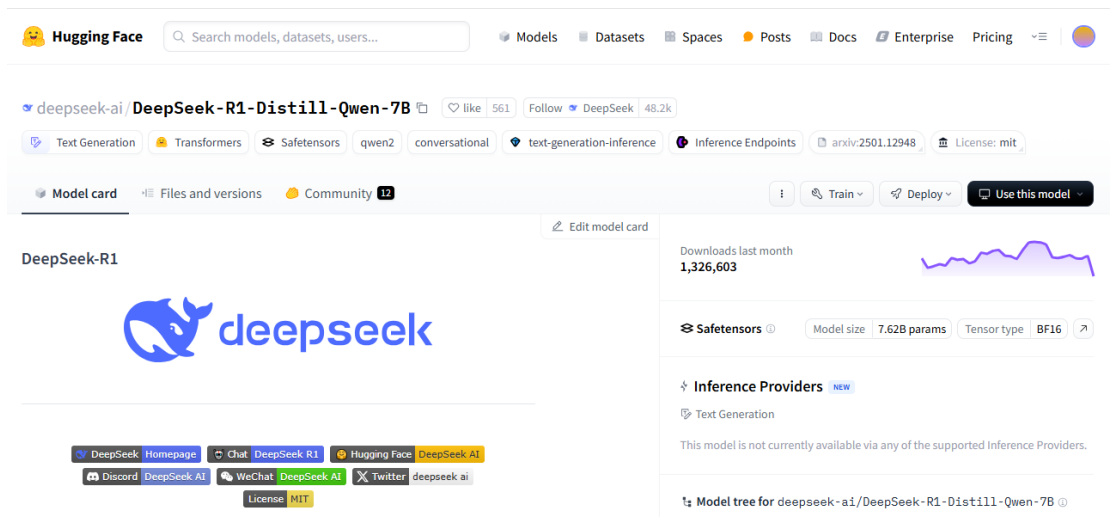


Figura 3-7 Repositorio en Hugging Face de la versión de DeepSeek usada (elaboración propia).

El lanzamiento de este modelo ha supuesto un antes y un después en el proceso de evolución de los LLMs ya que es capaz de lograr rendimientos iguales a modelos de pago como ChatGPT 4, o incluso superiores como en tareas de razonamiento lógico y matemático, pero con costos de desarrollo significativamente menores. Motivos por los que ha sido seleccionado como modelo para este proyecto [104].

3.2 Arquitectura del proyecto

Una vez enumeradas los recursos físicos y software que se van a usar, en esta sección se estudia la arquitectura y desarrollo del proyecto. En primer lugar, se introducirá la arquitectura del sistema presentando el modelo de bloques o módulos que se ha seguido para organizar todos los subprocesos involucrados en la ejecución del programa. Posteriormente, se estudia en detalle cada bloque, analizando los requisitos para llevarse a cabo, el código generado para su funcionamiento y se explica qué se genera en cada bloque para poder seguir con el siguiente.

Estudiando la Figura 3-8, su primer nodo muestra el paso inicial o paso cero para poder llevar a cabo el análisis de las encuestas. Se trata de la carga en el sistema de las encuestas cumplimentadas y escaneadas, que debe seguir ciertos criterios para el correcto funcionamiento del sistema. En primer lugar, el escaneado debe ser hecho siguiendo en todo momento el mismo método, como puede ser la misma resolución o los mismos PPI, y no utilizar varios procesos para escanear las distintas encuestas. Para ello, se creará un directorio con el nombre “*Scanned_PDF*” que servirá de repositorio para todas las encuestas. Dentro de esta carpeta, podrán guardarse todos los grupos y subgrupos de carpetas, con libertad a la hora de nombrarlos, donde se organizarán los distintos grupos que han cumplimentado la encuesta. El programa está configurado para recibir dentro de la carpeta *Scanned_PDF* tantos grupos como el encuestador quiera dividirlos. Las encuestas escaneadas irán guardadas dentro de las carpetas de los subgrupos. Es recomendable usar un criterio de nombramiento de carpetas, siguiendo estructuras claras y simples para facilitar la comprensión de los elementos que se irán creando de manera automática.

Después, se observa que el proyecto se ha dividido en seis bloques secuenciales de trabajo. Son seis módulos en los que se llevan a cabo distintos subprocesos que generan unos resultados. Las salidas generadas por un bloque servirán como entrada para el bloque siguiente. Así, con los dos primeros bloques se obtienen y organizan las respuestas de todas las encuestas que se han cargado en el sistema en formato PDF, utilizando para ello una herramienta OCR capaz de extraer texto de imágenes. Luego, esta salida se proporciona al LLM con el objetivo que localice aquellas respuestas seleccionadas por el encuestado y las apunte en un documento de texto (o documentos formato TXT). En el tercer bloque se recuperan los datos de los ficheros TXT creados por el LLM y se cargan en tablas de formato CSV,

donde se ordenan y realizan cálculos para su posterior análisis. El cuarto bloque es el encargado de crear las gráficas que mostrarán los resultados en el informe final junto a las conclusiones. Estas conclusiones, son generadas en el quinto bloque, donde se le entregan al LLM los datos procesados en el tercer bloque y se le pide que genere análisis de cada pregunta. Y, por último, el sexto bloque corresponde al módulo en el que se crean de manera automatizada los informes, incluyendo las preguntas, las gráficas con resultados y las conclusiones generadas por el LLM.

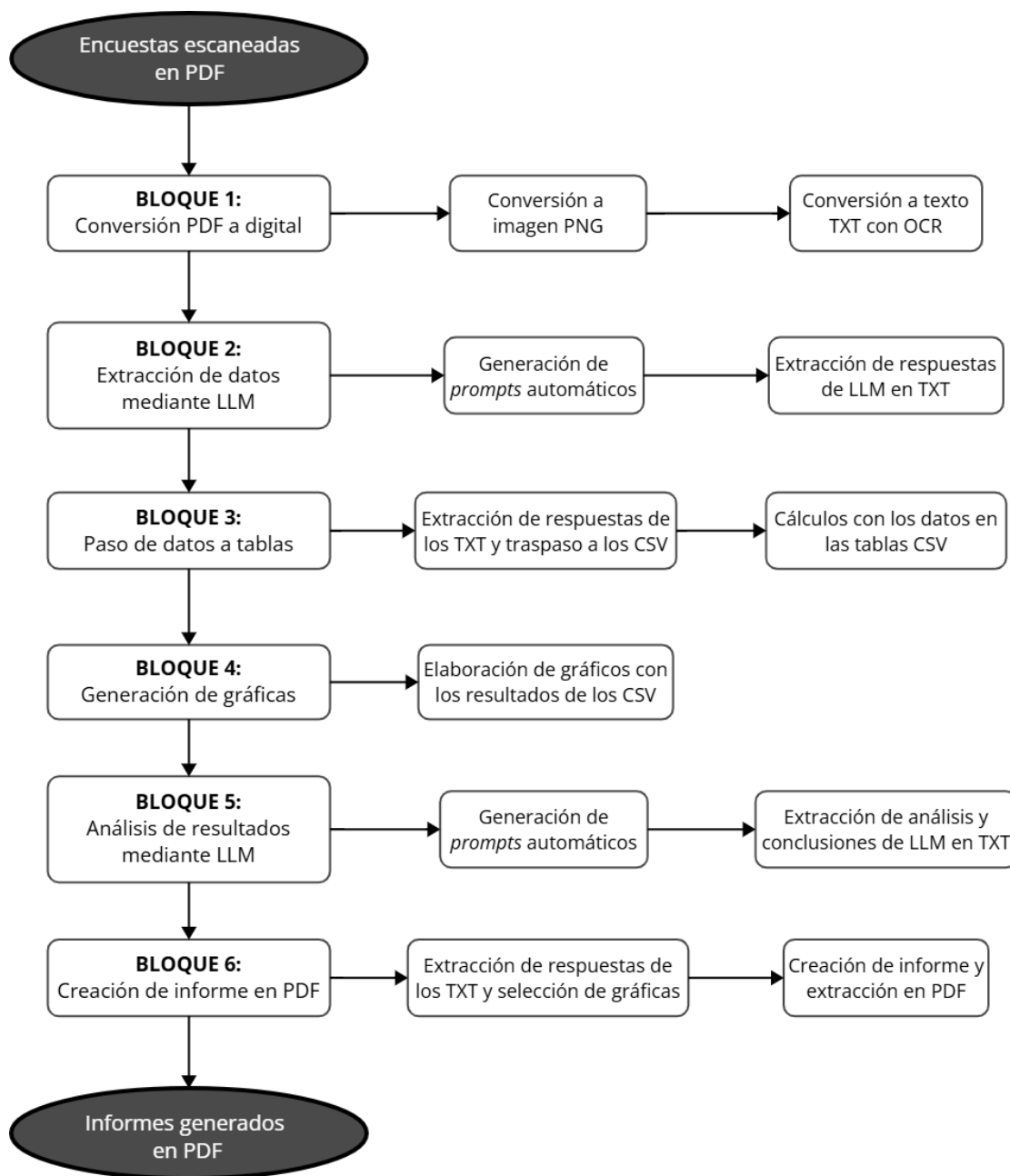


Figura 3-8 Esquema de la arquitectura del proyecto (elaboración propia).

Además, se debe escanear una copia de la encuesta sin cumplimentar, que servirá más adelante para que el LLM pueda realizar la primera de sus tareas. Esta encuesta sin cumplimentar, que llamaremos “encuesta original”, se guardará en un directorio más como los anteriores, respetando las carpetas de grupo y subgrupo, y asegurando que la carpeta de su grupo en el directorio *Scanned_PDF* sea la primera carpeta de la lista del directorio por defecto.

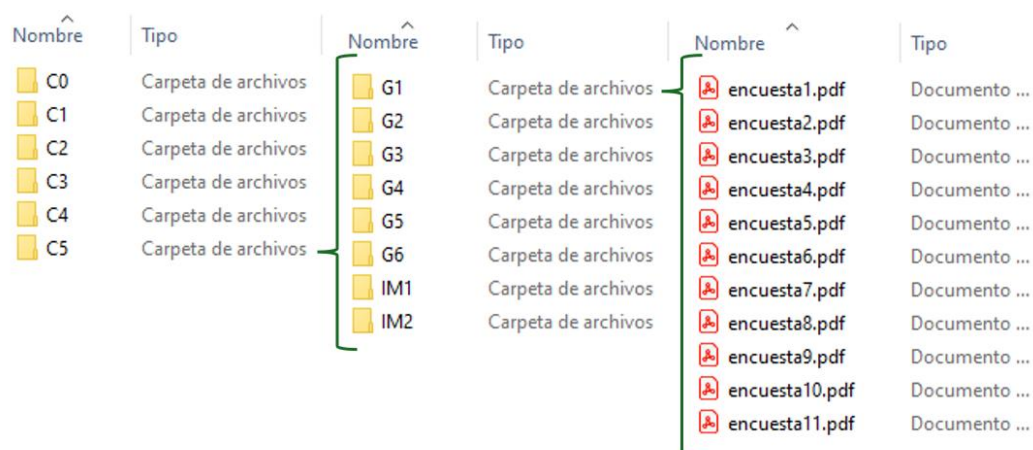


Figura 3-9 Desglose de encuestas de un grupo desde *Scanned_PDF* (elaboración propia).

Para ello, como se ve en la Figura 3-9, se distribuyeron copias que fueron cumplimentadas y entregadas y que han sido escaneadas siguiendo el mismo proceso en el mismo escáner. Una vez obtenidos todos los documentos PDF de las encuestas cumplimentadas se han almacenado en carpetas de nivel subgrupos correspondientes a los diferentes grupos de clase de cada curso: *G1*, *G2*, *IM*, etc.; y estos a su vez en carpetas de nivel grupo correspondientes a cada curso: *C1*, *C2*, etc. A continuación, se archivó una copia escaneada de la encuesta original en una carpeta llamada *G1*, respetando que siga los dos niveles de directorios; y esta, dentro de una carpeta llamada *C0*, respetando el requisito que dicta que la carpeta que contiene la encuesta original debe ser la primera en el directorio *Scanned_PDF*.

Para entender el desarrollo del proyecto y de cada bloque por separado, hay que entender cómo se ha desarrollado la arquitectura en los diferentes scripts de código. Como se ve en la Figura 3-10, el proyecto se compone de catorce ficheros de código: uno principal, seis correspondientes a los bloques, cinco auxiliares con las herramientas necesarias y dos correspondientes a los LLM.

Aux_CSV.py	Python File
Aux_graficas.py	Python File
Aux_informe.py	Python File
Aux_OCR.py	Python File
Aux_prompt.py	Python File
Bloque_1.py	Python File
Bloque_2.py	Python File
Bloque_3.py	Python File
Bloque_4.py	Python File
Bloque_5.py	Python File
Bloque_6.py	Python File
LLM_DeepSeek.py	Python File
LLM_Llama3.py	Python File
Main.py	Python File

Figura 3-10 Ficheros de código empleados (elaboración propia).

En primer lugar, el script llamado *Main.py*, visible en el Anexo III, hace la función de columna vertebral del sistema. Será el código que ejecutará el usuario una vez haya cargado en el sistema todos los documentos necesarios y de él serán llamados todos los diferentes bloques para asegurar una ejecución ordenada y clara en todo momento. Por cada bloque tiene una interacción de control para que el propio usuario le permita continuar o cancele la ejecución del programa. Para llamar a los bloques, concretamente al script conductor de cada uno, se utiliza el comando `subprocess.run` de la

biblioteca `subprocess`, como se ve en la Tabla 3-1, que ejecuta (*run*) el script seleccionado desde el sistema operativo. De tal manera que `Main.py` ejecuta y espera a que dicho proceso termine, momento en el que avisa de que ha terminado y solicita permiso para continuar con el siguiente.

```
print("\nBuenos días, bienvenido al programa de análisis automático de encuestas
académicas mediante el uso de LLM, ",end="")
comenzar_conversion = str(input("\ndesea comenzar? [y/n]: "))
while (comenzar_conversion != "y" and comenzar_conversion != "n"): comenzar_conversion =
str(input("\ndesea comenzar? [y/n]: "))
if comenzar_conversion == "n":
    print("\nHasta la próxima!\n")
    sys.exit()
else: print("\nCargando modelos de lenguaje...\n")

subprocess.run(["python3", "LLM_Llama3.py"])
subprocess.run(["python3", "LLM_DeepSeek.py"])

print("\nModelos cargados. Procedo con la conversión de los pdf a imágenes y texto.\n")

subprocess.run(["python3", "Bloque 1.py"])
```

Tabla 3-1 Código `subprocess` en el código `Main.py` (elaboración propia)

Cabe destacar que en el script `Main.py`, como se ve en la Tabla 3-1, el primer bloque se encarga de cargar los modelos de Llama3 y DeepSeek-R1 en el entorno. Se trata de los scripts `LLM_Llama3.py` y `LLM_DeepSeek.py` respectivamente, que se pueden estudiar en el Anexo VI. Cuando estos scripts son ejecutados, los LLM no realizan ningún proceso, únicamente se cargan en el sistema para que puedan ser llamados desde los bloques dos y cinco sin necesidad de volverlos a cargar. Una vez finalizada la carga de los modelos, el script `Main.py` procede a ejecutar el script del primer bloque, el fichero `Bloque_1.py`.

3.2.1 Bloque 1: conversión de información del PDF a formato TXT

El primer bloque del sistema, como se ha comentado anteriormente, se ejecuta el script `Bloque_1.py`, que podemos encontrar en el Anexo A4.1, en el que se procede a convertir todas las encuestas escaneadas en PDF a ficheros de texto, pasando antes por formato de imagen PNG, que es el que trabaja TesseractOCR.

En primer lugar, será necesario determinar si todas las caras de los documentos escaneados tienen texto, ya que, de no indicarse, el escáner crearía una última cara en blanco en el documento PDF. El problema, es que la herramienta OCR es sensible a ruido en las imágenes que analiza, por lo que, al no haber texto, el sistema trataría de analizar cualquier mancha o sombra en la imagen, generando un texto ficticio e incomprensible que provocaría problemas a posteriori. Por ello, el usuario tendrá que indicar el número de carillas que sí disponen de texto. De esta forma se asegura de que el OCR analice sólo las caras de los PDF que tienen texto y que solo extrae el texto correspondiente a esas caras.

A partir de ahí, el script `Bloque_1.py`, sirve de estructura “vertebral” del primer bloque, tiene por objetivo recorrer con bucles los directorios existentes para extraer todo lo necesario para realizar su cometido, creando nuevos directorios que almacenarán los resultados obtenidos siguiendo la misma estructura del directorio explorado. Concretamente, a partir de carpetas y archivos dentro del directorio *Scanned_PDF*, se procede a crear dos directorios principales con el comando *mkdir*, de la biblioteca *os*, a los que llama: *Encuestas_PNG* y *Encuestas_TXT*. Estos dos directorios se irán desarrollando internamente con la misma estructura que el directorio *Scanned_PDF*, manteniendo los nombres de las carpetas para mejor comprensión.

Una vez ambas creadas, se procede a utilizar la función *Pdfconverter* visible en la Tabla 3-2, del script *Aux_OCR.py*, que se encargará de procesar los documentos PDF. Esta función requiere:

- La ruta de la encuesta en PDF
- La ruta de la carpeta recién creada donde se guardarán las encuestas en PNG
- La ruta de la carpeta recién creada donde se guardarán las encuestas en TXT
- El nombre del documento a crear
- El número de caras con texto de la encuesta en PDF, introducido por el usuario
- Y, por último, el nombre del fichero de la encuesta en PDF

```
def Pdfconverter(ruta_inicial, ruta_final, ruta_txt_final, Nombre_doc, Npages,
Nombre_pdf):
    pages = convert_from_path(ruta_inicial, 500)
    if (Npages == 0): Npages = len(pages)
    i=0
    for page in pages:
        i=i+1
        nueva_ruta = ruta_final + "/" + str(Nombre_doc) + "_" + str(i) + ".png"
        page.save(nueva_ruta, 'PNG')
        print("    -->Convertido el documento: " + str(Nombre_pdf) + ".pdf en la imagen:
" + str(Nombre_doc) + "_" + str(i) + ".png",end="")
        if (i <= Npages):
            custom_config = r'--oem 3 --psm 6'
            extracted_text = pytesseract.image_to_string(nueva_ruta,
config=custom_config, lang="eng+spa")
            print(" y en el texto: " + str(Nombre_doc) + "_" + str(i) + ".txt")
            txt_name = ruta_txt_final + "/" + str(Nombre_doc) + "_" + str(i) + ".txt"
            with open(txt_name, "w", encoding="utf-8") as file:
                file.write(extracted_text)
```

Tabla 3-2 Función de conversión de PDF a TXT (elaboración propia).

Una vez dentro de la función, en primer lugar, convierte cada cara del PDF a una imagen PNG usando el comando *convert_from_path*, y las guarda en un que llama *pages*, que se irá recorriendo para hacer la conversión a texto. Después, en caso de que el usuario no introdujera número de caras, iguala el número de caras a convertir al total del documento, y, una vez hecho esto, procede con el bucle que irá convirtiendo cara a cara el documento PDF.

El bucle en cuestión guardará la imagen de cada una de las carillas del PDF almacenadas en el vector en la carpeta de imágenes. En caso de que sea una carilla con texto, procede a la extracción del texto mediante el OCR. Para ello, es necesario primero configurar los parámetros para la conversión, entre los que se incluye la indicación del lenguaje en el que se encuentran los documentos. En este caso se incluyó el español y el inglés: *lang="eng+spa"*, ya que algunas de las preguntas de la encuesta incluyen palabras en inglés, lo que provocaría errores en la conversión de no indicarse. Con el comando *pytesseract.image_to_string* se extrae el texto de la imagen y se guarda en el fichero TXT correspondiente.

Es importante recordar, que dentro del directorio *Scanned_PDF*, en la carpeta *C0* se encuentra la encuesta original que también es procesada, tarea clave para el siguiente bloque.

3.2.2 Bloque 2: extracción de respuestas del encuestado empleando LLM

En el segundo bloque, se recurre por primera vez a la asistencia de los modelos LLM, para que, dada una encuesta, realicen la extracción de las opciones seleccionadas por los encuestados. Estos resultados se almacenarán de nuevo en un fichero TXT en el que se llevará la cuenta.

Considerando que en este bloque se recurre a los LLM, no solo será necesario llamar a la función `analisis_LLM` de los ficheros de los LLM inicializados, sino que también es necesario automatizar la generación de *prompts*¹⁷, de tal manera que en cada momento el programa ajuste la petición a la necesidad del momento. Para ello, el fichero `Aux_prompt.py` tiene definidas cuatro funciones que sirven para generar los diferentes *prompts* siguiendo cuatro modelos definidos que serán usados a lo largo del sistema.

En este caso, el fichero conductor del bloque es el script `Bloque_2.py`, visible en el Anexo A4.2, que arranca inicializando las variables y el fichero `prompt.txt`, donde quedan guardados los *prompts* generados. Una vez inicializados, se recorren las carpetas del directorio `Encuestas_TXT`, donde se encuentran los documentos con las encuestas convertidas a TXT. A la vez, crea el directorio principal `Resultados_TXT`, donde crea carpetas siguiendo la estructura del directorio `Encuestas_TXT`, y donde se almacenarán los resultados extraídos de las encuestas cumplimentadas por los alumnos.

```
def prompt_encabezado(ruta_reference):
    pregunta = "In this text, there is a questionnaire consisting of questions and
    corresponding answers. I need you to clearly distinguish between questions and their
    respective answer choices. List all answer choices using the following format:
    '1a;1b;2a;2b;2c;3a;...;' where each number represents the question number, and each
    letter represents a specific answer choice within that question. Additionally, for every
    question, include an extra answer choice labeled with the letter 'z'. You must add this
    'z' option even to those questions that originally do not have multiple-choice answer
    options. Please present everything in a single line. The text is: \n"

    with open(ruta_reference, "r", encoding="utf-8") as file:
        texto = file.read()
        prompt = pregunta + texto

    ruta_archivo = r"Recursos/prompt.txt"

    with open(ruta_archivo, "w", encoding="utf-8") as file:
        file.write(prompt)
```

Tabla 3-3 Código generación de *prompt* para extraer encabezado (elaboración propia).

Además, se crea el documento `encabezado.txt` que almacenará un recopilatorio del número de preguntas y el número de posibles respuestas a cada pregunta, de forma que en el futuro se pueda generar tablas que servirán en el siguiente bloque del sistema. Con este fin, se recurre al LLM, llamando a la función `prompt_encabezado`, visible en la Tabla 3-3, que se encuentra en el fichero `Aux_prompt.py`. Esta función sirve para generar el *prompt* que da instrucciones a la LLM para realizar el mencionado recopilatorio con el formato indicado. Crea para ello una introducción con instrucciones y lo concatena con el texto de la encuesta original. Finalmente lo guarda en el archivo `prompt.txt` para ser usado en la siguiente función de `Bloque_2.py`.

Una vez creado el *prompt*, se llama a la función `analisis_LLM` con el *prompt* creado, como vemos en la Tabla 3-4, para obtener y almacenar la secuencia de letras y números que se convertirán en los encabezados de las tablas CSV. En el siguiente bloque del sistema.

¹⁷ Un *prompt* es una instrucción o entrada textual que se proporciona a un modelo de lenguaje para que genere una respuesta o realice una tarea específica.

```

ruta_encabezado = r"Recursos/encabezado.txt"
with open(ruta_encabezado, "w", encoding="utf-8") as file:
    pass
for p in range(Noriginales): #Aquí estoy creando el txt del encabezado para usar luego
    en las tablas CSV
    prompt_encabezado(rutas_originales[p]) #genera prompt para sacar encabezado del
    original
    respuesta_encabezado = analisis_LLM(ruta_prompt) #recibe respuesta de encabezado
    del original
    with open(ruta_encabezado, "a", encoding="utf-8") as file:
        file.write(respuesta_encabezado) #escribe en el txt de encabezado
    print("Creado encabezado sin novedad, guardado en " + ruta_encabezado)
    
```

Tabla 3-4 Código de secuencia de generación de encabezado (elaboración propia).

Una vez realizada la tarea anterior, se procede a la tarea principal del bloque: el análisis de las encuestas para la extracción de las respuestas. Para ello el script localiza cada archivo de texto de la encuesta a analizar y el fichero de texto de la carilla correspondiente de la encuesta original. Estas dos rutas se entregan como parámetro a la función `prompt_extracción`, que podemos ver en la Tabla 3-5. En este caso, las instrucciones del *prompt* que se genera en esta función son mucho más extensas y detalladas. En ellas, se le explica al LLM que va a recibir dos textos, el original y otro con errores, y que tiene que ser capaz de distinguir los errores, es decir, las respuestas de cada encuestado, y generar una respuesta perfectamente formalizada siguiendo un determinado criterio para que después puedan ser tratadas en el siguiente bloque.

```

def prompt_extraccion(ruta_reference, ruta_error):

    prompt1 = """I will provide you with two texts. The first one, which I will call
    REFERENCE_TEXT, will contain the text you must use as a reference. The second one will be
    called ERROR_TEXT and contains a very similar text but with some differences. It is a
    text composed of questions numbered with numbers and answers labeled with letters. Your
    goal is to detect, for each question number, which answer subsection contains the error.
    The only errors you must consider are those within the answers. There can be more than
    one subsection with an error per question. Only changes in the first word of each answer
    subsection should be considered as errors. You must not mark as an error any changes
    within the content of the subsection, even if there are spelling errors, typographical
    mistakes, or modifications anywhere else within the text that is not its first word. The
    first word of the subsection is understood to be the identifier (for example: 'a)', 'b)',
    etc.). As long as the identifier is correct, there will be no error. Do not disregard
    changes to special characters such as '@' or '('. The only character you must disregard
    if detected is the character '|'. That is, if you detect the character '|', ignore it as
    if it weren't there. You must not mark as an error any changes occurring after the words
    "Otro/s" and "Otra/s", regardless of their nature. This means any modification, addition,
    or deletion of characters after these words must be completely ignored in your analysis,
    even if they involve special characters like '(', ')', '_', '-', or spaces. Remember that
    I'm only interested in errors in the first word of each answer subsection, as those are
    the ones you must indicate to me. If the answer identifier (example: 'b)') is correct and
    the change occurs in the word immediately following it, it will not be considered an
    error. However, if the identifier has been modified in any way (for example: 'c)')
    instead of 'c)'), that will indeed be considered an error. Remember not to consider as
    errors any spelling errors, typographical mistakes, or modifications in any part of the
    text other than its first word. For instance, if in a word that isn't the first word
    there is an error due to an accent mark, it will not be considered an error. For example:
    if instead of 'código' it says 'codigo', it will not be considered an error. Remember
    that if the first word is the subsection identifier and it has no errors, you must not
    count it as an error. For example: if the subsection begins with 'a) Sf. Antes', even if
    'Sf.' wasn't in the original text, since it doesn't alter the first word ('a)'), there's
    no error in that subsection.

    You will indicate the errors to me in the following format: 'x:y;x:y...', without
    line breaks or separation across multiple lines, such that 'x' corresponds to the
    question number, and 'y' to the letter (the subsection) of the response where you've
    detected the error. If there is more than one subsection with an error in a question,
    list all the erroneous subsections separated by commas. For example: '1:d;2:a;3:b,c'. In
    """
    
```

```

this example, an error was detected in subsection 'd' of question 1, an error in
subsection 'a' of question 2, and errors in subsections 'b' and 'c' of question 3. If no
errors are detected in a question, use the character 'z' as the value of 'y'. Therefore,
your response must consist solely of a single line of numbers and letters as I've
specified. I now proceed to provide the texts:
REFERENCE_TEXT: ""
    with open(ruta_reference, "r", encoding="utf-8") as file:
        REFERENCE_TEXT = file.read()

    prompt2 = ""
ERROR_TEXT: ""
with open(ruta_error, "r", encoding="utf-8") as file:
    ERROR_TEXT = file.read()

    prompt3 = ""

    Analyze the texts, look for errors, and give me the answer exactly as I've requested.
    Give me only the answer I've requested. Give me the answer in one line. Don't show me the
    prompt I've written.'""

    prompt = prompt1 + REFERENCE_TEXT + prompt2 + ERROR_TEXT + prompt3

    ruta_archivo = r"Recursos/prompt.txt"

    with open(ruta_archivo, "w", encoding="utf-8") as file:
        file.write(prompt)

```

Tabla 3-5 Código de generación de prompt para extracción de respuestas (elaboración propia).

Como se ve en la función, se concatenan las instrucciones con los textos del archivo de la encuesta original y el del archivo de texto de la encuesta a estudiar. Posteriormente, este nuevo texto se almacena en el archivo *prompt.txt*, para que pueda ser llamado posteriormente por el LLM. La respuesta del LLM se guarda automática y secuencialmente en el archivo TXT creado para almacenar las respuestas del LLM.

Este bloque de código está programado para recorrer todas las carpetas y subcarpetas, de tal manera que todas las encuestas queden revisadas y sus respuestas correctamente apuntadas. Una vez se han analizado todos los archivos del directorio *Encuestas_TXT*, se ejecuta el bloque tres.

3.2.3 Bloque 3: conversión de resultados a CSV y tratamiento de datos

El tercer bloque es el encargado de extraer las respuestas que ha señalado el LLM en el segundo bloque y rellenar cuidadosamente tablas en ficheros CSV para poder tratar los datos y así poder continuar con los siguientes bloques. El objetivo de este bloque es que calcule los números totales de respuestas de cada apartado en cada pregunta, ofreciendo resultados agrupados por cada grupo de clase, por cada curso y los resultados totales, es decir, de todos los cursos.

El fichero con el código conductor de este bloque es el script *Bloque_3.py*, visible en el Anexo A4.3. La peculiaridad de este bloque es que se almacenan todos los pasos intermedios. Esto es, se almacenan los datos tratados en los primeros pasos para poder ser convertidos a tablas CSV más adelante, como por ejemplo la fusión de todos los datos generados para los diferentes grupos, que se guardan para la tabla de cursos, y así sucesivamente. Por lo tanto, a lo largo del script se crean una serie de directorios, como *Grupos_CSV* o *Cursos_CSV*, para almacenar las tablas con los resultados ya sumados, y directorios como *Total_preCSV_TXT* para almacenar los pasos intermedios en formato TXT.

Para ello, en el fichero *Bloque_3.py*, se carga el contenido de *encabezado.txt*, creado en el bloque anterior, y, posteriormente, a medida que se vaya recorriendo la estructura del directorio *Resultados_TXT*, se irán guardando las tablas de datos.

Como se ve en la Tabla 3-6, se recuperan las rutas de los TXT y se entregan a la función `tabla_csv_grupo`, que genera la tabla CSV correspondiente, usando de cabecera el contenido de *encabezado.txt*, y completando los datos con las respuestas extraídas por el LLM. Por ejemplo, si en la tercera encuesta del grupo estudiado, se han marcado las respuestas ‘a’ y ‘d’ de la quinta pregunta, en la tabla CSV, en la tercera fila, grabará un 1 en las columnas ‘5a’ y ‘5d’. Por último, crea una nueva fila debajo en la que se hace sumatorio de todas las superiores, quedando registrado cuántas veces se ha marcado cada apartado de cada pregunta en el grupo analizado. Finalmente, gracias a la función `txt_sum_grupo`, se extraen los valores de la primera fila (el encabezado) y de la última (el sumatorio), y se concatenan columna a columna para guardarlos en los archivos de texto de los pasos intermedios.

```
ruta_txt_grupo = "Resultados_TXT/" + cursos[x] + "/" + txt_resultados[y] + ".txt"
#recuperamos la ruta del txt a procesar
ruta_resultados_grupo = "Grupos_CSV/" + cursos[x] + "/" + txt_resultados[y] + ".csv"
#creamos la ruta del csv donde irá la tabla de resultados procesados
# Los CSV se crean directamente en la función
tabla_csv_grupo(ruta_encabezado, ruta_txt_grupo, ruta_resultados_grupo) #Se llama a
función para procesar. Se le entrega la ruta del txt donde está el encabezado de las
columnas de la tabla, la ruta del txt a procesar y la del csv de resultados procesados
print("\n Terminado el grupo " + txt_resultados[y] + ".\n")

resultados = txt_sum_grupo(ruta_resultados_grupo)
pre_resultados = txt_resultados[y] + ";"
with open(ruta_resultados_resumen_txt, "a", encoding="utf-8") as file:
    file.write("\n" + pre_resultados + resultados)
print("Guardado en " + ruta_resultados_resumen_txt)
```

Tabla 3-6 Secuencia de generación de tablas CSV (elaboración propia).

Una secuencia parecida sucede para crear las tablas en los niveles superiores empleando la función `recopilatorio_datos`. En este caso, en lugar de recurrir a los archivos de texto creados por el LLM recurre a los archivos de pasos intermedios creados en el paso anterior. Una vez localizados procede a generar tablas CSV para los niveles superiores, que se guardarán en el directorio correspondiente.

Terminada la tarea de creación de tablas de resultados por niveles, el sistema procede a ejecutar el cuarto bloque del proyecto.

3.2.4 Bloque 4: creación de gráficas de los resultados

El cuarto bloque es el encargado de crear las gráficas correspondientes a las respuestas de cada pregunta, partiendo de los resultados procesados en el bloque anterior relativos a los tres niveles: grupos de clase, cursos y conteo total. Con estos datos se puede proceder a realizar gráficas de cada uno de estos niveles.

Analizando el script `Bloque_4.py`, disponible en el Anexo A4.4, volvemos a recorrer los directorios correspondientes a las tablas de resultados tratados anteriormente: *Grupos_CSV*, *Cursos_CSV* y *Total_CSV*; y de nuevo se creará una estructura similar para almacenar las gráficas de cada uno de los niveles. Se crea además el fichero *Tabla_transición.csv*, que servirá de transición entre los CSV de resultados y la generación de gráficas.

Para lograr el cometido, se recurre, en primer lugar, a la función, *transformar_csv*, que sirve para extraer las respuestas de todos los ficheros CSV y darles el formato necesario para generar gráficas y guardarlos en el fichero *tabla_transición.csv*. Dicho fichero dispone de tres columnas, donde la primera es el número de la pregunta, la segunda es el apartado de esa pregunta, y la tercera es la cantidad de veces que ha sido respondida, habiendo una fila para cada apartado de cada pregunta.

En segundo lugar, se recurre a la función `generar_graficas`, visible en la Tabla 3-7, que será la encargada de crear las gráficas de barras y circulares (pastel) que serán incluidas en el informe final. A esta función se le entregan como parámetros la tabla CSV donde se encuentran los datos almacenados y la ruta de la carpeta donde debe guardar las gráficas.

```
def generar_graficas(csv_path, save_path):

    df = pd.read_csv(csv_path, delimiter=';', header=None) # Cargar el archivo CSV con el
    delimitador ';'
    df.columns = ['Pregunta', 'Opción', 'Valor'] # Asignación de nombres de columnas
    manualmente

    df['Valor'] = pd.to_numeric(df['Valor'], errors='coerce') # convierte la columna de
    valores a numérico

    preguntas = df['Pregunta'].unique() #agrupa por la primera columna
    for pregunta in preguntas:
        sub_df = df[df['Pregunta'] == pregunta]

        #GRÁFICO DE BARRAS:
        plt.figure(figsize=(8, 6))
        cmap = plt.get_cmap("Blues") #asignar colores según valores
        norm = plt.Normalize(sub_df['Valor'].min(), sub_df['Valor'].max())
        colores = [cmap(0.5 + 0.5 * norm(v)) for v in sub_df['Valor']]

        plt.bar(sub_df['Opción'], sub_df['Valor'], color=colores)
        plt.xlabel('Opción', fontsize=20, fontname='serif')
        plt.ylabel('Valor', fontsize=20, fontname='serif')
        plt.title(pregunta, fontsize=30, fontname='serif')
        plt.xticks(fontsize=16, fontname='serif')
        plt.yticks(fontsize=16, fontname='serif')
        plt.grid(axis='y', linestyle='--', alpha=0.7)

        save_file_bar = os.path.join(save_path, f"{pregunta}_barras.png") # Guarda el
        gráfico de barras
        plt.savefig(save_file_bar, bbox_inches='tight')
        plt.close()
```

Tabla 3-7 Código de generación de gráficas (elaboración propia).

Finalmente, se crean las gráficas. Estas pueden ser de barras o circulares. En el caso de los histogramas se tienen que establecer ciertos parámetros: el tamaño, los colores de las barras y la distribución de colores. En este proceso se utilizaron tonos de azul, proporcionando un tono más oscuro cuando se trataba del campo más numeroso. Después se establecen el resto de los parámetros como la fuente y el tamaño de letra utilizados. Finalmente se guarda el gráfico en la ruta dada como parámetro. En el caso de las gráficas circulares, se especifica que solo se representen valores superiores a cero y la gama de colores a utilizar. Se establecen los parámetros correspondientes como fuente y tamaño de letra y se guarda la gráfica generada en la ruta dada.

Una vez finalizada la tarea de elaboración de todas las gráficas se procede al quinto bloque.

3.2.5 Bloque 5: análisis de resultados mediante LLM

El quinto bloque, como se especifica en la introducción de la arquitectura, también requiere recurrir al LLM para realizar la tarea. Continuando con el estudio de resultados a tres niveles, se le encarga realizar análisis de las respuestas de cada pregunta y de generar una conclusión, para adjuntarlos en el informe final junto con los gráficos.

El fichero que conduce este bloque es el script `Bloque_5.py`, disponible en el Anexo A4.5. Este script, al igual que los anteriores, recorre la carpeta de resultados del bloque anterior para tratar sus datos y generar nuevos resultados, realizando primero el análisis por grupos, después por cursos y terminando con el análisis de los datos totales. En este caso se generan tres pares de directorios. Esto se debe a que, manteniendo el análisis de respuestas a los tres niveles, hay que crear directorios para almacenar archivos

TXT de recopilatorio: “*Respuestas_Grupos_txt*”, “*Respuestas_Cursos_txt*” y “*Respuestas_Total_txt*”; y directorios para almacenar los documentos TXT que almacenen los análisis de cada pregunta y las conclusiones que saque para poder extraerlas al realizar los informes. Estos directorios son: “*Conclusiones_Grupos_LLM*”, “*Conclusiones_Cursos_LLM*” y “*Conclusiones_Total_LLM*”.

```
def prompt_analisis(ruta_preguntas, ruta_respuestas, num_linea):

    prompt1 = "For the next question: "
    with open(ruta_preguntas, "r", encoding="utf-8") as file:
        for i, linea in enumerate(file, start=0):
            if i == num_linea:
                PREGUNTA = linea.strip() # Devuelve la línea sin saltos de línea
    prompt2 = "'\n\nThe answers received are: "
    with open(ruta_respuestas, "r", encoding="utf-8") as file:
        for i, linea in enumerate(file, start=0):
            if i == num_linea:
                RESPUESTAS = linea.strip() # Devuelve la línea sin saltos de línea
    prompt3 = "'\n\nPrepare an analysis in Spanish of those answers and give it to me.
    Just give me the analysis in one paragraph. Don't show me the prompt I've written."
    prompt = prompt1 + PREGUNTA + prompt2 + RESPUESTAS + prompt3
    ruta_archivo = r"Recursos/prompt.txt"
    with open(ruta_archivo, "w", encoding="utf-8") as file:
        file.write(prompt)
```

Tabla 3-8 Código para crear *prompt* de generación de análisis de preguntas (elaboración propia).

En el script encargado de llevar a cabo este bloque se recorren las carpetas de CSV creados en el tercer bloque, recuperando las rutas de cada archivo CSV y se generan las rutas de los archivos TXT que recopilan los resultados y se inicializan. Posteriormente, se recurre a `transformar_csv_a_txt`, función de `Aux_CSV.py`, a la que se entregan las dos rutas mencionadas y que extrae los datos del encabezado y de la fila sumatorio, creando secuencias formalizadas que sean fáciles de entender por el LLM. Después, se inicializa el archivo TXT en el que se irán apuntando los análisis de cada pregunta. Para ello, se procede al bucle en el que recorren el archivo de la encuesta original junto con los archivos de respuestas recién creados, línea a línea, y se crea un nuevo *prompt* con la función `prompt_analisis` que vemos en la Tabla 3-8, en el que se genera un *prompt* que presente la pregunta que se está analizando con sus apartados, y posteriormente las respuestas elegidas para cada apartado, solicitando al LLM mediante la función `analisis_LLM` previamente estudiada que redacte un análisis en castellano que explique las respuestas de dicha pregunta. La respuesta de dicha interacción con el LLM es el párrafo de análisis de la pregunta que se está estudiando.

Los análisis de las preguntas se van grabando secuencialmente en el TXT de análisis y conclusión, una vez ha terminado de analizar todas las preguntas de cierto CSV, procede a solicitar una conclusión al LLM reuniendo todos los análisis que ha generado anteriormente. Para ello genera un nuevo *prompt* con la función `prompt_conclusion`, visible en Tabla 3-9 y también en el script `Aux_prompt.py`, que procede a leer todos los análisis del documento TXT que se está generando, y le da instrucciones al LLM para que sintetice todos los análisis y genere un párrafo de conclusión en castellano. Se recurre de nuevo a la función `analisis_LLM`, y su respuesta, el párrafo de conclusión, se añade al documento TXT de análisis y conclusión, en la última fila de este, para ser usado en el siguiente bloque.

```
def prompt_conclusion(ruta_analisis):

    prompt1 = "I am going to provide you with the analyses in Spanish of the different
responses from a survey. I want you to read everything and give me a final analysis in
the form of a conclusion that summarizes the general ideas expressed in the survey
responses. I want the final conclusion on the survey's topic of study to be clearly
stated and explained. The analyses, which are numbered according to each question, are as
follows: \n\n"

    with open(ruta_analisis, "r", encoding="utf-8") as file:
        PREGUNTA = file.read()

    prompt2 = "'\n\nI want you to generate only one paragraph in Spanish with the
complete conclusion analysis."

    prompt = prompt1 + PREGUNTA + prompt2
    ruta_archivo = r"Recursos/prompt.txt"
    with open(ruta_archivo, "w", encoding="utf-8") as file:
        file.write(prompt)
```

Tabla 3-9 Código para crear *prompt* de generación de conclusión (elaboración propia).

Una vez generados todos los archivos de texto que contienen los análisis de las respuestas y las conclusiones, quedan finalizadas todas las tareas de preparación del informe, dando paso al último bloque.

3.2.6 Bloque 6: generación automática de informes en PDF

El sexto bloque, último del proyecto, es el encargado de reunir toda la información textual y gráfica generada en los bloques cuatro y cinco, con el fin de generar un informe final en PDF que analice los resultados de la encuesta. El fichero que ejecuta el bloque 6, el script `Bloque_6.py`, visible en el Anexo A4.6, ofrece, en primer lugar, la opción al usuario de seleccionar sobre qué niveles de análisis quiere hacer informes. Es decir, se ofrecen las opciones de generar los informes por grupos de clase, por cursos, o un informe total. Además, se ofrece una cuarta opción que es realizar informes por cursos y el informe total, ya que son los niveles que ofrecen visiones más amplias y útiles.

Una vez elegido el nivel de análisis del que se van a realizar las encuestas, se solicita al usuario que elija el formato de las gráficas que quiere que consten en el informe: gráficas de barras o circulares (de pastel). A continuación, se recorren las carpetas para recuperar las rutas de las gráficas y de los análisis y conclusiones correspondientes.

Para la realización del PDF, se recurre a la función `generar_pdf` del fichero auxiliar `Aux_informe.py`, visible en el Anexo A5.5. A esta función hay que proporcionarle como parámetros: título del informe, ruta a la encuesta original, ruta al directorio de las gráficas correspondiente, ruta al documento TXT donde se encuentran los análisis y la conclusión correspondiente, ruta de los PDF que se van a generar, y, por último, el formato de gráficas que ha elegido el usuario.

Por lo tanto, en `Bloque_6.py` se crea el directorio *Informes_PDF* donde se almacenarán todos los informes que el usuario haya solicitado, y se recorren simultáneamente los directorios de gráficas y de documentos TXT de análisis y conclusiones, de donde se van extrayendo las rutas de una en una. Una vez tiene todas las rutas extraídas, procede a llamar a la función `generar_pdf`, entregando todos los parámetros de manera automatizada. En el caso del título del informe, es un parámetro que, si se están generando informes de los niveles de grupos de clase y cursos, incluye el nombre de la asignatura del curso en el que se han realizado las encuestas analizadas; y, en el caso de que sea el nivel de grupos de clase, el grupo de clase en ese curso. Por ello se crea un vector que tenga en cada posición los nombres de las asignaturas de manera ordenada, como se ve en la Tabla 3-10.

```
asignatura = ["Informática para la Ingeniería", "Fundamentos de Electrotecnia",
"Tecnología Electrónica", "Fundamentos de Automática", "Fundamentos de Redes de
Ordenadores"]
titulo_informe = "Informe de Encuestas 2025 - " + asignatura[a]
```

Tabla 3-10 Creación de vector con nombres de asignaturas (elaboración propia).

Una vez llamada, comienza la generación del informe en el fichero `Aux_informe.py`. Lo primero que destaca en este fichero es la importación de los módulos de la biblioteca *reportlab*, que proporcionan las herramientas para generar el PDF. Después, únicamente encontramos la función `generar_pdf`. Esta comienza generando y configurando el archivo PDF, estableciendo tanto el formato de hoja como otros valores como el ancho de margen, espacio entre bloques, tamaño de las imágenes, etc. Una vez configurado, genera el encabezado de la primera cara del informe, donde inserta los escudos de la ENM y del CUD-ENM a cada lado y escribe en medio el título del informe. Separada por una línea negra horizontal del encabezado, comienza la estructura del informe, que se basa en una construcción por bloques repetitiva, en la que cada bloque lo componen:

- Número de pregunta y pregunta correspondiente en la parte superior del bloque.
- Gráfica del formato elegido de la pregunta en cuestión debajo a la izquierda.
- Apartados de la pregunta en cuestión a la derecha de la gráfica.
- Análisis del LLM de los resultados de esa pregunta debajo de la imagen y apartados.

Esta estructura se repite una vez por cada pregunta que compone la encuesta. Y, por último, añade un bloque final que titula “Conclusiones del análisis” en la que incluye la conclusión generada por el LLM con los análisis de las respuestas de la encuesta. Una vez rellenado el informe, procede a guardarlo en la carpeta correspondiente del directorio *Informes_PDF*.

3.2.7 Códigos de implementación de los LLMs

El script `LLM_Llama3.py`, disponible en el Anexo A6.2, es el encargado de inicializar el modelo de LLM que posteriormente se usará. En este sentido, comienza ofreciendo la posibilidad de usar cuantificación en el proceso. Esto significa que se reduce la precisión de los pesos del modelo (por ejemplo, de 32 bits flotantes a 8 o 4 bits) con el objetivo de reducir el tamaño del modelo, acelerar su inferencia y disminuir el uso de memoria, sin perder demasiado rendimiento. En el caso del script `LLM_Llama3.py`, en caso de elegir cuantificación se reduciría a 4 bits. Se recomienda utilizar cuantificación, por ejemplo, en equipos que no tengan CPU.

```
def analisis_LLM (ruta_prompt):
    with open(ruta_prompt, "r", encoding="utf-8") as file:
        prompt = file.read()

    response = text_generator(prompt)
    respuesta = response[0]['generated_text']
    return respuesta
```

Tabla 3-11 Función de llamada al modelo Llama3 (elaboración propia).

Una vez elegido, se carga el modelo y queda listo para su uso gracias a la función definida `analisis_LLM`, como se ve en la Tabla 3-11, que recibe como parámetro la ruta del documento *prompt.txt*, que será creado en el directorio *Recursos* para generar en él el *prompt* que se usará en cada interacción con el LLM. La función abre dicho fichero TXT y copia su contenido en la variable *prompt*, que se entrega a la función `text_generator` previamente creada y cargada, que hace de puerta del LLM y devuelve, a través de la función `response`, una respuesta que se devuelve al programa que llamó a `analisis_LLM` originalmente.

En el caso del script `LLM_DeepSeek.py`, disponible en el Anexo A6.1, no se da la opción de cuantificar porque, como se ha tratado previamente, ya es un modelo optimizado y no necesita simplificarse más. Por ello, se carga directamente en el sistema. Después, al igual que en el modelo anterior, vemos la función `analisis_LLM` en la Tabla 3-12, que recibe como parámetro la ruta del documento `prompt.txt`. Lo primero, será leer el fichero y extraer el contenido a la variable `prompt`, que pasa por la función `tokenizer` generando el `prompt` “tokenizado”, y de ahí, pasa a la función `model.generate`, que teniendo en cuenta todos los parámetros establecidos, genera una respuesta para ese `prompt` tokenizado. Finalmente, en la variable `response` se guarda la respuesta decodificada mediante `tokenizer.decode` y la devuelven al `prompt` donde se llamó a la función `analisis_LLM` originalmente.

```
def analisis_LLM(ruta_prompt):

    with open(ruta_prompt, "r", encoding="utf-8") as file:
        prompt = file.read()

    inputs = tokenizer(
        prompt,
        return_tensors="pt",
        padding=True,
        truncation=True
    ).to(model.device)
    with torch.no_grad():
        outputs = model.generate(
            inputs["input_ids"],
            attention_mask=inputs["attention_mask"],
            max_new_tokens=1000, # Limita los tokens utilizados
            temperature=0.2, # Controla creatividad (mayor, más creatividad)
            para evitar alucinaciones
            top_p=0.5, # Reduce diversidad, es más directo con su respuesta
            pad_token_id=tokenizer.eos_token_id,
            eos_token_id=tokenizer.eos_token_id, # Forzar fin de generación
        )

    response = tokenizer.decode(outputs[0], skip_special_tokens=True)
    conclusion = response.split("</think>", 1)[-1].strip() # Tomamos lo que
    está después y eliminamos espacios

    return conclusion
```

Tabla 3-12 Función de llamada al modelo de DeepSeek (elaboración propia).

Es destacable del modelo de DeepSeek que en todas sus respuestas muestra el razonamiento que ha realizado y posteriormente, la respuesta final. Sin embargo, para el sistema únicamente es útil la respuesta, por ello la respuesta del modelo pasa por un filtro en el que se edesprecia todo lo que no estuviera después de la palabra “</think>”. Una vez filtrado, devuelve la respuesta al bloque.

4 RESULTADOS OBTENIDOS

En el cuarto capítulo, se analizan todos los resultados que se han ido obteniendo de cada proceso, de tal manera que se estudie la viabilidad de cada uno y del conjunto. Primero, se presenta el conjunto de datos sobre los que trabajaremos durante el desarrollo y del que se han obtenido los resultados. Para presentar los resultados, se dividen todos los procesos y bloques en tres. En primer lugar, se exponen los resultados de todos los procesos desde la recepción de las encuestas en PDF hasta la extracción de los resultados por el LLM en el bloque 2. Después, se analizan los resultados de los siguientes procesos hasta la entrada de los análisis de resultados y de las conclusiones por parte del LLM en el bloque 5. Finalmente, se exponen los resultados del bloque 6, mostrando los informes generados en PDF como producto final.

4.1 Caso de uso: encuesta realizada en Fundamentos de Redes de Ordenadores

Para la ejecución de este proyecto, se han realizado encuestas en formato papel al alumnado durante el curso 2024-2025, resultando un total de trescientos sesenta y ocho encuestados. Se trata de una encuesta enmarcada en el proyecto de innovación docente que se está realizando en el CUD-ENM titulado “*Evaluación del uso de herramientas de IA Generativa como apoyo al proceso de aprendizaje de habilidades prácticas de ingeniería*”, que pretende evaluar el uso por parte de los alumnos de los LLM para su integración en su metodología de estudio y trabajo; y la percepción que tienen sobre el desarrollo de esta tecnología en su futuro profesional como oficiales de la Armada, entre otros. Para hacer el análisis de resultados, se toma como muestra las encuestas recogidas a los alumnos de quinto año, que realizaron las encuestas mientras cursaban la asignatura de Fundamentos de Redes de Ordenadores. Las preguntas de la encuesta se muestran en la Tabla 4-1.

- | |
|--|
| <ol style="list-style-type: none"> 1. ¿Has asistido a la conferencia de “Introducción al uso ético, responsable, crítico y transparente de la IA generativa” impartida a tu brigada? <ol style="list-style-type: none"> a. Si b. No 2. ¿Con qué frecuencia utilizas herramientas de inteligencia artificial generativa para tus estudios? <ol style="list-style-type: none"> a. Muy frecuentemente (4 o más veces por semana). b. Frecuentemente (1-3 veces por semana). c. Ocasionalmente (1-3 veces al mes). d. Raramente (menos de una vez al mes). e. Nunca. 3. ¿Cuál es el principal propósito por el que utilizas herramientas de IA generativa en tus estudios? <ol style="list-style-type: none"> a. Generación o lluvia de ideas. b. Ayuda con la redacción, traducción o edición de textos. |
|--|

- c. Resolución de problemas o dudas.
 - d. Creación de contenido visual o multimedia.
 - e. Otro (_____).
 - f. No las utilizo.
4. En caso de haberlas usado, ¿en qué medida crees que el uso de herramientas de IA generativa ha mejorado tu rendimiento académico?
- a. Ha mejorado significativamente mi rendimiento.
 - b. Ha mejorado ligeramente mi rendimiento.
 - c. No ha tenido impacto en mi rendimiento.
 - d. Ha empeorado ligeramente mi rendimiento.
 - e. Ha empeorado significativamente mi rendimiento.
 - f. No las he utilizado lo suficiente para opinar.
5. ¿Cuál consideras que es el mayor beneficio de utilizar herramientas de IA generativa en tus estudios? Selecciona todas las opciones que sean de aplicación.
- a. Ahorro de tiempo.
 - b. Mejora en la calidad de los trabajos.
 - c. Ayuda para entender conceptos complejos.
 - d. Acceso a información actualizada.
 - e. Otro (_____).
 - f. No veo beneficios significativos.
6. Si has hecho uso de herramientas de IA generativa, ¿para qué tipo de contenido las has empleado? Selecciona todas las opciones que sean de aplicación.
- a. Generación de texto
 - b. Generación de imágenes
 - c. Generación de audio
 - d. Generación de vídeo
 - e. Generación de código
 - f. Tareas exclusivas de revisión de la calidad lingüística de un texto o de traducción
 - g. Otro/s (_____)
7. Si has hecho uso de herramientas de IA generativa, ¿cuáles son las que has empleado?
- a. ChatGPT
 - b. DALL-E
 - c. Midjourney
 - d. Gemini
 - e. Copilot
 - f. Otra/s (_____)
 - g. No utilizo ninguna
8. Antes del presente proyecto de innovación docente (incluida la conferencia formativa previa a la brigada), ¿habías recibido alguna formación o guía sobre cómo utilizar éticamente las herramientas de IA generativa en tus estudios?
- a. Sí, he recibido formación completa.
 - b. He recibido alguna información, pero no formación completa.
 - c. No he recibido ninguna formación o guía.
9. ¿Cuál sería tu valoración general de la experiencia del proyecto de innovación docente?
- a. Muy positiva
 - b. Positiva
 - c. Indiferente
 - d. Negativa
 - e. Muy negativa
10. ¿Consideras que el haber participado en el proyecto de innovación docente cambiará tus hábitos de uso de herramientas de inteligencia artificial generativa?

	a. Sí. Antes no las usaba, y ahora pasaré a hacerlo de forma ética, crítica, responsable y transparente. b. Sí. Ya las usaba, pero ahora pasaré a hacerlo de forma ética, crítica, responsable y transparente. c. No. Ya las usaba de forma ética, crítica, responsable y transparente. d. No. No las usaba y seguiré sin hacerlo.
11.	¿Considerarías importante para la formación de un Oficial de la Armada incluir entre los contenidos del grado formación teórico-práctica en herramientas de inteligencia artificial? a. Muy importante. b. Importante. c. Indiferente. d. Poco importante. e. Nada importante.
12.	¿Te gustaría recibir formación adicional en este ámbito? a. Sí b. No
13.	Si tienes alguna sugerencia o comentario, puedes incluirlo en el cuadro de texto a continuación.

Tabla 4-1 Encuesta repartida al alumnado (elaboración propia).

El OCR utilizado no tiene la capacidad de realizar reconocimiento de texto manuscrito, por lo tanto, para este proyecto no se tienen en cuenta las preguntas en las que el encuestado escribe su respuesta abierta. En el caso de esta encuesta, en las preguntas 3, 5, 6 y 7, si el encuestado escogiera la opción en la que contempla otras opciones diferentes a las que ya aparecen en el resto de los resultados, ofreciendo un hueco al encuestado para introducir a mano su respuesta, se tendrá en cuenta que hay otras opciones consideradas por los encuestados, pero no es capaz de detectar cuáles. En el caso de la pregunta 13, que es de respuesta libre, aparecerá en todos los no contestada por los encuestados por ese motivo.

4.2 Resultados de la extracción de respuestas

En el primer tramo estudiado, se produce la conversión de los PDF a imágenes y estos a su vez a texto mediante el OCR. Finalmente se estudian los resultados del LLM en su proceso de extracción de respuestas de los textos creados.

4.2.1 Resultados de la conversión de PDF a PNG

Como se puede observar en la Figura 4-1, las imágenes muestran las encuestas escaneadas con todos los defectos característicos del escaneado de hojas: ruido en forma de sombras alrededor de los textos y en los márgenes de las hojas. También en la figura, se observa cómo la encuesta original, a la izquierda, y las encuestas cumplimentadas, a la derecha, han seguido el mismo proceso de escaneado, de tal manera que los errores que arrastren los siguientes procesos debido a las imperfecciones del escaneado serán similares y no supondrá un problema decisivo.

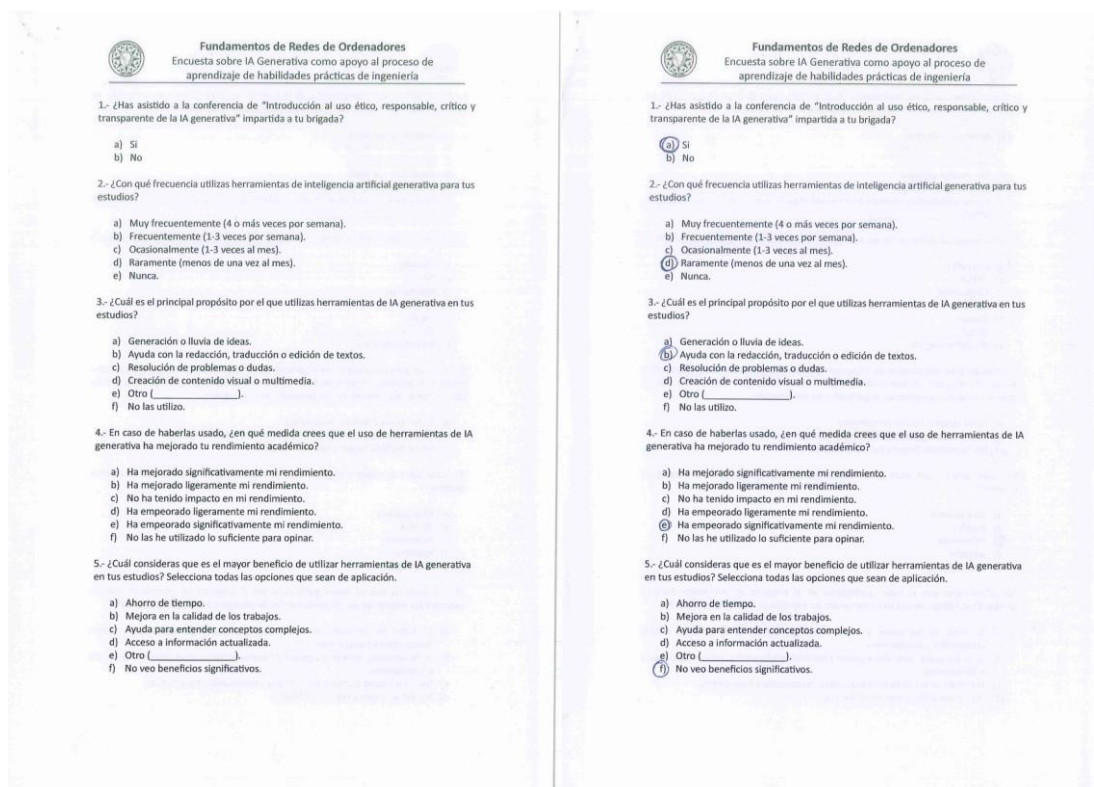


Figura 4-1 Imágenes producto de la conversión de PDF a PNG (elaboración propia).

Añadir que, los documentos de las encuestas escaneadas se componen de cuatro caras, de las cuales sólo hay preguntas y respuestas en tres, generando una cuarta imagen que está en blanco. Por lo tanto, se generan con este proceso un total de mil cuatrocientas setenta y dos imágenes correspondientes a las cuatro caras de todas las encuestas escaneadas y cuatro más correspondientes a la encuesta original.

4.2.2 Resultados del OCR.

En la Figura 4-2 observamos el resultado correspondiente de la conversión de imagen PNG a documentos de texto, empleando la biblioteca pytesseract, de la primera cara de la encuesta original.

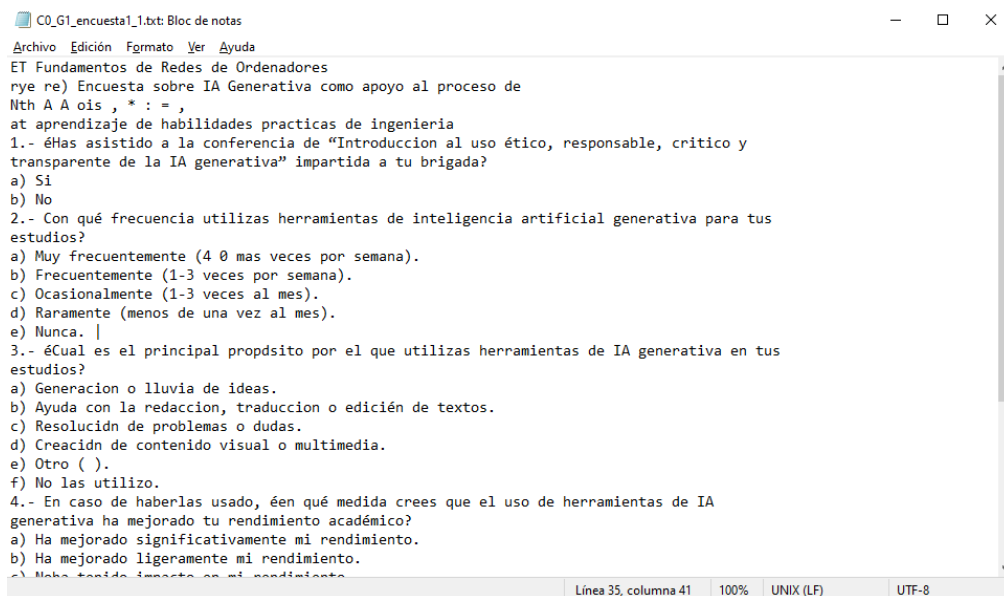


Figura 4-2 Texto extraído de la encuesta original (elaboración propia).

Una vez se compara el texto generado de la Figura 4-2 con las imágenes del escaneado de la Figura 4-1, es apreciable la falta de exactitud de la extracción de caracteres del OCR. Los ejemplos más destacables son en el texto correspondiente a la cabecera de la encuesta, donde el OCR genera secuencias de caracteres incoherentes como: “Nth A A ois , * : = ,” que complican el posterior análisis del texto.

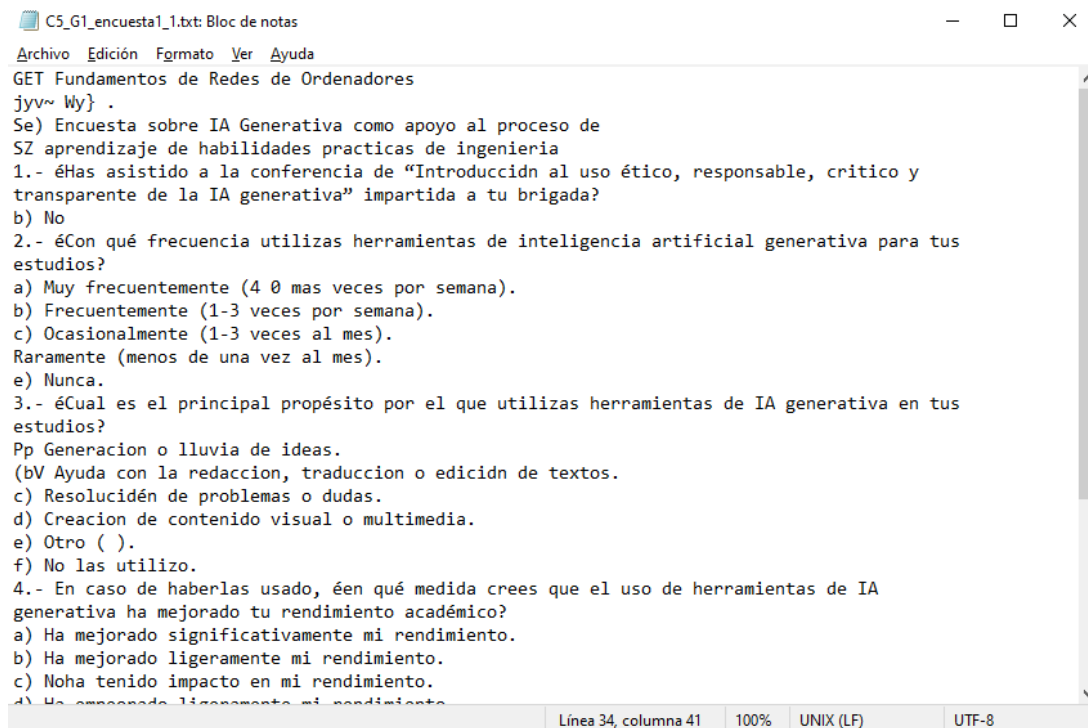


Figura 4-3 Texto extraído de una encuesta del quinto curso (elaboración propia).

En la Figura 4-3 se observa el resultado de la extracción de texto mediante el OCR de una encuesta cumplimentada. Si se compara con la Figura 4-2, que hace de referencia al ser la encuesta original, podemos observar que se mantienen errores en el escaneado, como los de la cabecera u otros como confundir el carácter ‘¿’ con el carácter ‘é’ al inicio de las preguntas. Dejando estos errores de lado, ya que ocurren también en el original, observamos la que fue la primera de las contrariedades que han surgido al desarrollar este proyecto: no hay ningún círculo rodeando el apartado marcado como vemos en las encuestas cumplimentadas.

Sin embargo, estudiando el texto generado es fácil darse cuenta de que, al inicio de algunos apartados de cada pregunta, la letra de numeración se ve claramente alterada. Comparando con la encuesta cumplimentada de la que ha sido extraído este texto, nos damos cuenta de que los apartados que muestran la alteración en los puntos de numeración coinciden con los apartados que han sido marcados por el encuestado. Por lo tanto, para los procesos de análisis de texto, se detecta que el conteo de respuestas de las encuestas se hará localizando los errores en las numeraciones de los apartados de cada pregunta.

Habrá que tener en cuenta que, en caso de que un encuestado marque una opción y, cambiando de opinión, tache la errónea para marcar la opción que considera correcta, ambos apartados darán error en el procesado OCR dejando errores en la numeración de los dos apartados y siendo ambos previsiblemente detectados como respuestas marcadas por el LLM. Por lo tanto, los resultados del sistema no se pueden considerar totalmente precisos.

4.2.3 Resultados del script creador de prompts

El fichero `Aux_prompt.py` no cuenta con ninguna biblioteca de funciones para llevar a cabo su tarea, pero no hace que este deje de ser fundamental. Como ya se vio en el capítulo de desarrollo cuenta

con cuatro funciones diferentes para crear cuatro modelos de *prompt* automatizados con los que recurrir al LLM para que genere el análisis y la respuesta deseados.

En este tramo de procesos, es necesario emplear un *prompt* para extraer el encabezado de las futuras tablas y otro *prompt* para localizar y hacer listado de las respuestas marcadas de cada encuesta. Por ello, genera *prompts* que da las instrucciones pertinentes en inglés para que el modelo las entienda sin problema y no falle a la hora de cumplir con los requisitos.

En cuanto al script para extraer las respuestas de las encuestas, se utilizan los textos extraídos mediante OCR correspondientes a la encuesta original y los de la encuesta cumplimentada de la que se quiere extraer los resultados, que se entregan a la función *prompt_extraccion*, obteniendo el texto de la Tabla 4-2.

"I will provide you with two texts. The first one, which I will call *REFERENCE_TEXT*, will contain the text you must use as a reference. The second one will be called *ERROR_TEXT* and contains a very similar text but with some differences. It is a text composed of questions numbered with numbers and answers labeled with letters. Your goal is to detect, for each question number, which answer subsection contains the error. The only errors you must consider are those within the answers. There can be more than one subsection with an error per question. Only changes in the first word of each answer subsection should be considered as errors. You must not mark as an error any changes within the content of the subsection, even if there are spelling errors, typographical mistakes, or modifications anywhere else within the text that is not its first word. The first word of the subsection is understood to be the identifier (for example: 'a'), 'b'), etc.). As long as the identifier is correct, there will be no error. Do not disregard changes to special characters such as '@' or '('. The only character you must disregard if detected is the character '|'. That is, if you detect the character '|', ignore it as if it weren't there. You must not mark as an error any changes occurring after the words "Otro/s" and "Otra/s", regardless of their nature. This means any modification, addition, or deletion of characters after these words must be completely ignored in your analysis, even if they involve special characters like '(', ')', '_', '-', or spaces. Remember that I'm only interested in errors in the first word of each answer subsection, as those are the ones you must indicate to me. If the answer identifier (example: 'b') is correct and the change occurs in the word immediately following it, it will not be considered an error. However, if the identifier has been modified in any way (for example: 'c') instead of 'b'), that will indeed be considered an error. Remember not to consider as errors any spelling errors, typographical mistakes, or modifications in any part of the text other than its first word. For instance, if in a word that isn't the first word there is an error due to an accent mark, it will not be considered an error. For example: if instead of 'código' it says 'codigo', it will not be considered an error. Remember that if the first word is the subsection identifier and it has no errors, you must not count it as an error. For example: if the subsection begins with 'a) Sf. Antes', even if 'Sf.' wasn't in the original text, since it doesn't alter the first word ('a)'), there's no error in that subsection.

You will indicate the errors to me in the following format: 'xy;xy...', without line breaks or separation across multiple lines, such that 'x' corresponds to the question number, and 'y' to the letter (the subsection) of the response where you've detected the error. If there is more than one subsection with an error in a question, list all the erroneous subsections separated with the number of their question. For example: '1d;2a;3b;3c'. In this example, an error was detected in subsection 'd' of question 1, an error in subsection 'a' of question 2, and errors in subsections 'b' and 'c' of question 3. If no errors are detected in a question, use the character 'z' as the value of 'y'. Therefore, your response must consist solely of a single line of numbers and letters as I've specified. I now proceed to provide the texts:

REFERENCE_TEXT: 'ee

6.- Si has hecho uso de herramientas de IA generativa, ¿para qué tipo de contenido las has empleado? Selecciona todas las opciones que sean de aplicacion.

a) Generacion de texto

b) Generacion de imagenes

c) Generacion de audio

d) Generacion de video

e) Generacion de cddigo

f) Tareas exclusivas de revision de la calidad lingiiistica de un texto o de traduccion

g) Otro/s ()

[...]

10.- ¿Consideras que el haber participado en el proyecto de innovación docente

cambiara tus habitos de uso de herramientas de inteligencia artificial generativa?

a) Si. Antes no las usaba, y ahora pasaré a hacerlo de forma ética, critica,

responsable y transparente.

b) Si. Ya las usaba, pero ahora pasaré a hacerlo de forma ética, critica, responsable y transparente.

c) No. Ya las usaba de forma ética, critica, responsable y transparente.

d) No. No las usaba y seguiré sin hacerlo.

,

ERROR_TEXT: 'ome

ts ie

ees

ee os ee

6.- Si has hecho uso de herramientas de IA generativa, ¿para qué tipo de contenido las has empleado? Selecciona todas las opciones que sean de aplicadn.

(a) Generación de texto

b) Generacion de imagenes

c) Generacion de audio

d) Generacidén de video

e) Generacidn de código

6) Tareas exclusivas de revision de la calidad lingtiistica de un texto o de traduccién

g) Otro/s (__)

[...]

10.- ¿Consideras que el haber participado en el proyecto de innovacdn docente cambiara tus habitos de uso de herramientas de inteligencia artificial generativa?

a) Si. Antes no las usaba, y ahora pasaré a hacerlo de forma ética, critica, responsable y transparente.

b) Si. Ya las usaba, pero ahora pasaré a hacerlo de forma ética, critica, responsable y transparente.

c) No. Ya las usaba de forma ética, critica, responsable y transparente.

d) No. No las usaba y seguiré sin hacerlo.

,

Analyze the texts, look for errors, and give me the answer exactly as I've requested. Give me only the answer I've requested. Give me the answer in one line. Don't show me the prompt I've written. Just show me the answer."

Tabla 4-2 Prompt generado para buscar las respuestas (elaboración propia).

Con este *prompt* se observa cómo se establecen las instrucciones especificando todas las pautas o normas que debe cumplir el modelo y posteriormente se introducen los dos textos a comparar detallando cuál es el de referencia y cual contiene errores. De esta forma el LLM debe realizar un listado con un formato establecido de las respuestas, es decir, los errores en la numeración, detectados en los apartados de cada pregunta. También es importante reseñar que en todas las preguntas se añade un apartado que sumará en el caso de que la pregunta no haya sido contestada por el encuestado, importante en la pregunta 13 que quedará señalada automáticamente en todas las encuestas analizadas.

También en este tramo, también se recurre al LLM para generar un encabezado para las tablas que se generan con el siguiente bloque. Para ello recurre al TXT generado de la conversión de la encuesta original y le solicita a la LLM que extraiga el número de preguntas y de apartados siguiendo un formato especificado, como se observa en la Tabla 4-3.

"In this text, there is a questionnaire consisting of questions and corresponding answers. I need you to clearly distinguish between questions and their respective answer choices. List all answer choices using the following format: '1a;1b;2a;2b;2c;3a;...;' where each number represents the question number, and each letter represents a specific answer choice within that question. Additionally, for every question, include an extra answer choice labeled with the letter 'z'. You must add this 'z' option even to those questions that originally do not have multiple-choice answer options. Please present everything in a single line. The text is:

LET Fundamentos de Redes de Ordenadores

rye 5) Encuesta sobre IA Generativa como apoyo al proceso de

T= aprendizaje de habilidades prácticas de ingeniería

1.- ¿Has asistido a la conferencia de "Introducción al uso ético, responsable, crítico y transparente de la IA generativa" impartida a tu brigada?

a) Si

b) No

2.- ¿Con qué frecuencia utilizas herramientas de inteligencia artificial generativa para tus estudios?

a) Muy frecuentemente (4 o más veces por semana).

b) Frecuentemente (1-3 veces por semana).

c) Ocasionalmente (1-3 veces al mes).

d) Raramente (menos de una vez al mes).

e) Nunca. |

3.- ¿Cuál es el principal propósito por el que utilizas herramientas de IA generativa en tus estudios?

a) Generación o lluvia de ideas.

b) Ayuda con la redacción, traducción o edición de textos.

c) Resolución de problemas o dudas.

d) Creación de contenido visual o multimedia.

e) Otro ().

f) No las utilizo.

4.- En caso de haberlas usado, ¿en qué medida crees que el uso de herramientas de IA generativa ha mejorado tu rendimiento académico?

a) Ha mejorado significativamente mi rendimiento.

b) Ha mejorado ligeramente mi rendimiento.

c) No ha tenido impacto en mi rendimiento.

d) Ha empeorado ligeramente mi rendimiento.

e) Ha empeorado significativamente mi rendimiento.

f) No las he utilizado lo suficiente para opinar.

5.- ¿Cuál consideras que es el mayor beneficio de utilizar herramientas de IA generativa en tus estudios? Selecciona todas las opciones que sean de aplicación.

a) Ahorro de tiempo.

b) Mejora en la calidad de los trabajos.

c) Ayuda para entender conceptos complejos.

d) Acceso a información actualizada. y

e) Otro ().

f) Nuevos beneficios significativos. |'''

Tabla 4-3 Prompt para realizar el encabezado de las tablas (elaboración propia).

La última pregunta de prueba para el modelo no cuantificado de Llama3 es la misma que en el caso anterior, pero se solicita que devuelva únicamente los caracteres erróneos. Sin embargo, como se ve en la Figura 4-6, se limita a analizar los errores, pero no es capaz de devolver una respuesta formalizada. Además, en algunas de las interacciones con la misma pregunta da los fallos característicos de repetición inexplicable, como se ve en la respuesta anterior de la misma figura.

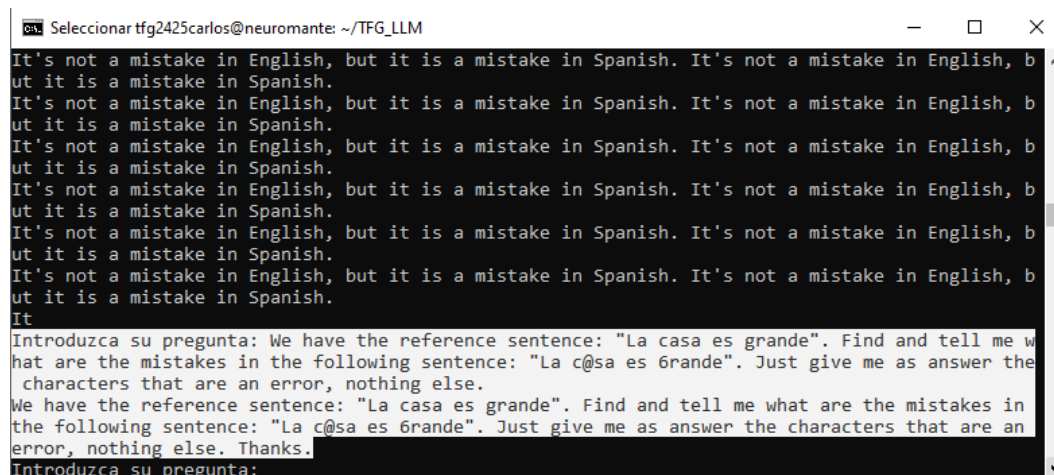


Figura 4-6 Respuesta formalizada por Llama3 cuantificado (elaboración propia).

A la versión de Llama3 sin cuantificación, se le hacen las mismas preguntas para poder comparar resultados. En primer lugar, se le pregunta la capital de Estados Unidos. Como se ve resaltado en la Figura 4-7, el modelo no cuantificado es capaz de generar un texto bien estructurado y que responde la pregunta de una forma perfectamente eficiente. Sin embargo, ya destaca que como respuesta genera el mismo texto de manera repetida varias veces, lo que puede suponer un problema.

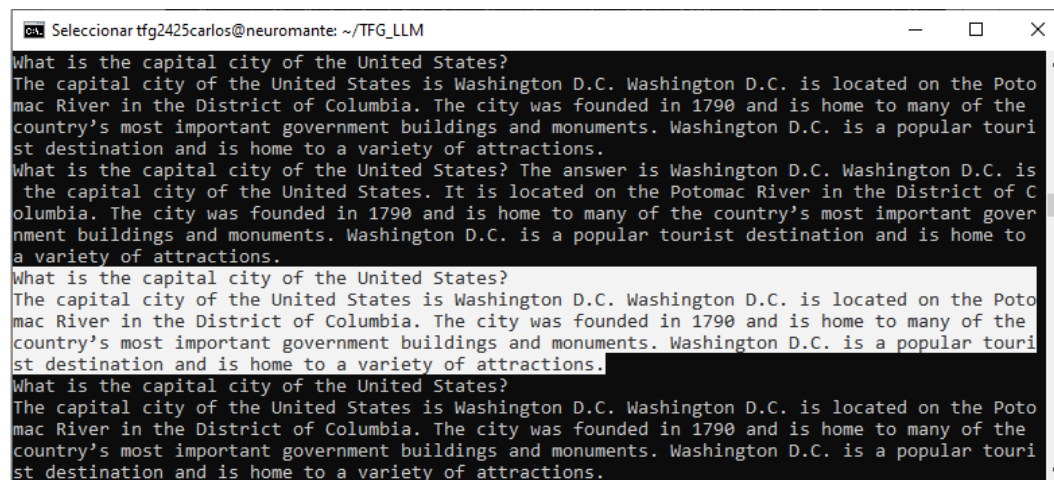
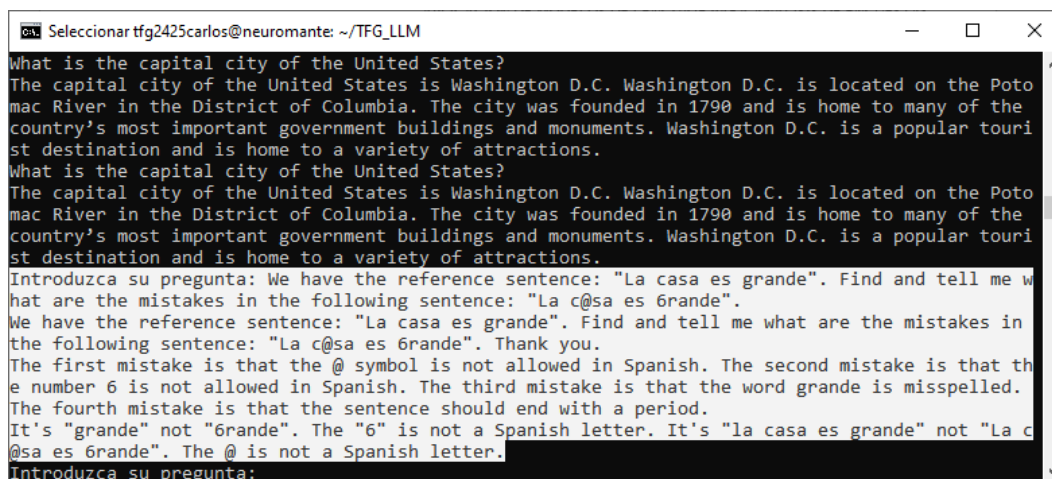


Figura 4-7 Pregunta simple a Llama3 no cuantificado (elaboración propia).

A continuación, se le ofrecen las mismas frases a comparar que al modelo cuantificado para pedirle que localice y devuelva los errores. Como se ve en la Figura 4-8, el modelo es capaz de localizar los fallos de caracteres en la frase con errores.

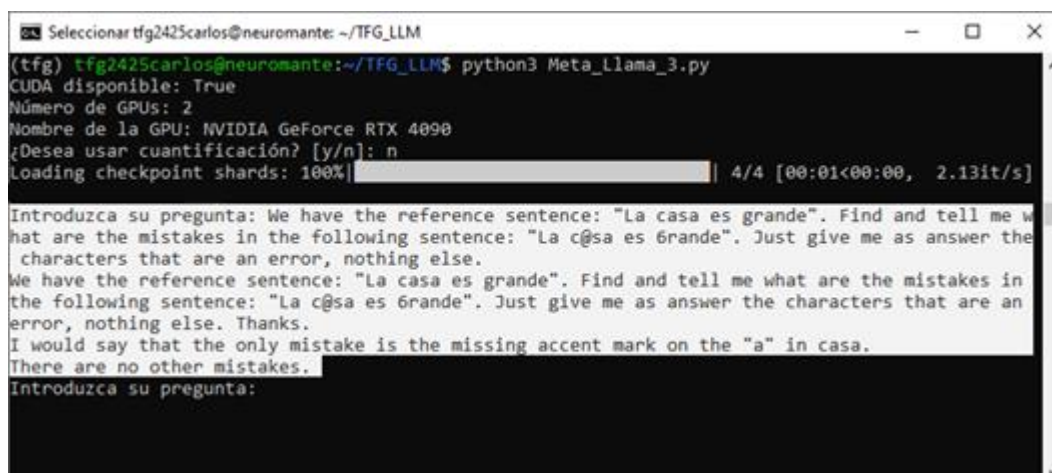


```

Seleccionar tfg2425carlos@neuromante: ~/TFG_LLM
What is the capital city of the United States?
The capital city of the United States is Washington D.C. Washington D.C. is located on the Potomac River in the District of Columbia. The city was founded in 1790 and is home to many of the country's most important government buildings and monuments. Washington D.C. is a popular tourist destination and is home to a variety of attractions.
What is the capital city of the United States?
The capital city of the United States is Washington D.C. Washington D.C. is located on the Potomac River in the District of Columbia. The city was founded in 1790 and is home to many of the country's most important government buildings and monuments. Washington D.C. is a popular tourist destination and is home to a variety of attractions.
Introduzca su pregunta: We have the reference sentence: "La casa es grande". Find and tell me what are the mistakes in the following sentence: "La c@sa es 6grande".
We have the reference sentence: "La casa es grande". Find and tell me what are the mistakes in the following sentence: "La c@sa es 6grande". Thank you.
The first mistake is that the @ symbol is not allowed in Spanish. The second mistake is that the number 6 is not allowed in Spanish. The third mistake is that the word grande is misspelled. The fourth mistake is that the sentence should end with a period.
It's "grande" not "6grande". The "6" is not a Spanish letter. It's "la casa es grande" not "La c@sa es 6grande". The @ is not a Spanish letter.
Introduzca su pregunta:
    
```

Figura 4-8 Localización de errores en texto por Llama3 no cuantificado (elaboración propia).

La última pregunta de prueba para el modelo no cuantificado de Llama3 es la misma que en el caso anterior, pero se solicita que devuelva únicamente los caracteres erróneos. Sin embargo, como se ve en la Figura 4-9, se limita a analizar los errores, pero no es capaz de devolver una respuesta formalizada.



```

Seleccionar tfg2425carlos@neuromante: ~/TFG_LLM
(tfg) tfg2425carlos@neuromante:~/TFG_LLM$ python3 Meta_llama_3.py
CUDA disponible: True
Número de GPUs: 2
Nombre de la GPU: NVIDIA GeForce RTX 4090
¿Desea usar cuantificación? [y/n]: n
Loading checkpoint shards: 100% | 4/4 [00:01<00:00, 2.13it/s]
Introduzca su pregunta: We have the reference sentence: "La casa es grande". Find and tell me what are the mistakes in the following sentence: "La c@sa es 6grande". Just give me as answer the characters that are an error, nothing else.
We have the reference sentence: "La casa es grande". Find and tell me what are the mistakes in the following sentence: "La c@sa es 6grande". Just give me as answer the characters that are an error, nothing else. Thanks.
I would say that the only mistake is the missing accent mark on the "a" in casa.
There are no other mistakes.
Introduzca su pregunta:
    
```

Figura 4-9 Respuesta formalizada por Llama3 no cuantificado (elaboración propia).

Es de importancia mencionar también que, a diferencia del modelo cuantificado, Llama3 no cuantificado tarda un mucho tiempo en realizar los análisis. Esto es un factor limitante cuando se van a realizar numerosas interacciones ya que el tiempo acumulado en los análisis puede ser demasiado alto.

Para el modelo de DeepSeek, se realizan de nuevo las tres preguntas para conocer sus capacidades. Cuando se le pregunta cuál la capital de Estados Unidos, como se ve en la Figura 4-10, entendemos el funcionamiento del LLM de DeepSeek, que devuelve todo el razonamiento realizado hasta llegar a la solución, que ofrece después de la palabra `</think>`. La respuesta en este caso es correcta, y en la mayoría de las interacciones con este LLM las respuestas son válidas, pero hay otras en las que el modelo alucina o devuelve respuestas con más información de la pedida.

```

Seleccionar tfg2425carlos@neuromante: ~/TFG_LLM
I should also consider if there are any alternative names for Washington, D.C. I think sometime
s people refer to it as just Washington, but the full name is more precise. So in the answer, I
should probably include both the short name and the full name to cover all bases.

Another thing to think about is why someone would ask this question repeatedly. Maybe they're p
reparing for a test, or perhaps they're part of a larger process where this information is need
ed multiple times. Regardless, providing the correct and concise answer each time is key.

I also wonder if there's any regional variation or if someone might confuse the capital with an
other city, like Atlanta or Birmingham. But no, Washington, D.C. is the official capital, so th
at's clear.

In summary, each of these questions is asking for the same information, so the answer is consis
tent every time. It's a simple question, but it's good to practice answering it correctly, espe
cially if someone is testing the ability to recall such facts.
</think>
The capital city of the United States is Washington, D.C.
Introduce pregunta:

```

Figura 4-10 Pregunta simple al modelo de DeepSeek (elaboración propia).

Pasando a la pregunta de comparación de textos, se le ofrecen las mismas frases que al modelo Llama3. Como vemos en la Figura 4-11, este modelo hace un análisis preciso y lo muestra los razonamientos que ha seguido de forma clara. Destacar la presencia de la palabra `</think>` para indiciar el inicio de la respuesta.

```

tfg2425carlos@neuromante: ~/TFG_LLM
I think that covers the mistakes and the reasons behind the word similarities. It's all about r
ecognizing how small changes in spelling or characters can lead to similar-sounding or similar-
looking words, which are often mistakes in typing or writing.
</think>

The sentence "La c@sa es 6grande" contains two main mistakes, identified through the concept of
Word Similarity:

1. ***"c@sa" instead of "casa"***:
   - The word "c@sa" is similar to "casa" because the '@' symbol replaces the second 'a' in "ca
sa". This is a common typo where an 'a' is mistakenly replaced with '@'.

2. ***"6grande" instead of "grande"***:
   - The word "6grande" is similar to "grande" because the number '6' resembles the letter 'g' i
n some fonts, especially lowercase. This is a substitution error where the 'g' was mistyped as
'6'.

In summary, the mistakes are due to typographical errors where similar-looking characters repla
ced the correct ones, leading to words that are similar in appearance but incorrect in meaning.

```

Figura 4-11 Localización de errores en texto por el modelo DeepSeek (elaboración propia).

Por último, se solicita que haga el mismo análisis comparativo pero que en esta ocasión se limite a enseñar los errores. En la Figura 4-12, tras la palabra `</think>` únicamente observamos una línea en la que indica cuáles son los errores. Sin haber dado más instrucciones sobre el formato, consideramos buena la respuesta al haber modificado su comportamiento habitual de respuestas extensas y explicativas por una respuesta clara y directa como se había solicitado.

```

tfg2425carlos@neuromante: ~/TFG_LLM
So, the two mistakes in the sentence are the use of '@' in "c@sa" and the use of '6' in "6 grande". Therefore, the characters that are errors are '@' and '6.'

I should make sure there aren't any other mistakes. Let me check the structure again. The sentence starts with "La," which is correct. Then "c@sa" instead of "casa." That's definitely a mistake because "casa" is a word that doesn't have an '@' in it. The rest of the sentence is "es 6 grande," which should be "es grande." So, yeah, the only errors are those two characters.

I don't think there are any other issues with the sentence. The punctuation is correct, it's a complete sentence, and the capitalization is right. So, yeah, I'm confident that the errors are just the '@' and the '6.'

**Final Answer**
The mistakes are the characters '@' and '6'.
</think>

The mistakes in the sentence "La c@sa es 6 grande" are the characters '@' and '6'.

Introduce pregunta:
    
```

Figura 4-12 Respuesta formalizada por el modelo DeepSeek (elaboración propia).

Tras el análisis de funcionamiento básico de los modelos LLM, podemos apoyarnos en la Tabla 4-4 para comparar las tres opciones y hacer una hipótesis previa sobre los posibles resultados una vez se le entregue el *prompt* generado para analizar las encuestas cumplimentadas con la encuesta original.

Modelo	Cuantificación	Razonamiento	Respuesta	Formalizado	Velocidad	Viabilidad
Llama3	Sí	Bueno	Errónea	Nulo	Alta	No
Llama3	No	Bueno	Buena	Nulo	Muy baja	No
DeepSeek	-	Bueno	Buena	Aceptable	Alta	Sí

Tabla 4-4 Comparación análisis básico de LLM (elaboración propia).

Comparando las distintas opciones se ha establecido que la única que tiene posibilidades de resultar viable es el modelo de DeepSeek. El modelo de Llama3 muestra fallos críticos tanto con cuantificación como con sin ella. En el caso del modelo cuantificado observa que la calidad del análisis es mala ya que se inventa los errores de las frases, y el modelo no cuantificado tarda demasiado en hacer un análisis. Además, ambos demuestran capacidad nula para adaptar el mensaje al formato solicitado en el *prompt*.

4.2.5 Resultados del análisis de encuestas por los LLMs

Habiendo estudiado por separado las capacidades básicas de análisis de texto de los LLM, se procede a realizar las pruebas de los modelos en la tarea de analizar las encuestas para extraer las respuestas. Pese a los resultados obtenidos en el apartado anterior, se realiza un estudio tanto de DeepSeek como del modelo Llama3 cuantificado y sin cuantificar.

Partiendo del *prompt* ya generado, y explicado en la sección 4.2.3, en el script *Bloque_2.py* se recurre a la función *analisis_LLM*, que quedó explicada en la sección 3.2.7 de arquitectura del sistema, con la que el *prompt* generado se entrega al modelo LLM para que genere su análisis y devuelva la respuesta. El resultado del script *Bloque_2.py* deberían ser los archivos TXT rellenos como la Figura 4-13. Sin embargo, ninguno de los resultados obtenidos por los modelos LLM son los esperados.

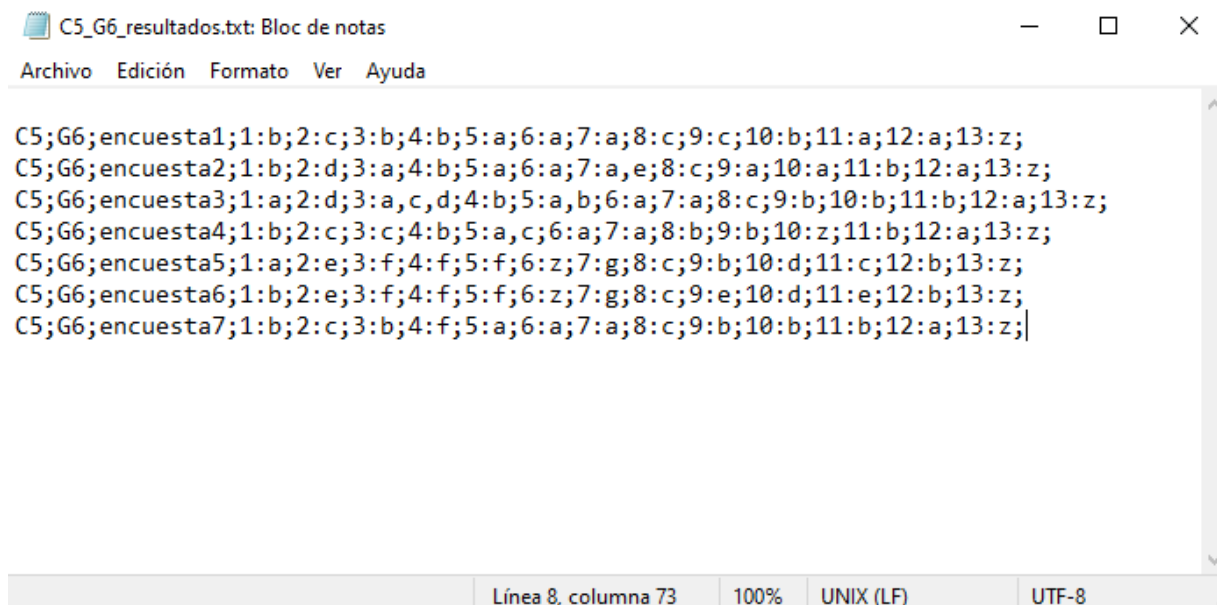


Figura 4-13 Archivo TXT con las respuestas de las encuestas de un grupo (elaboración propia).

Para el modelo Llama3, se realizan pruebas con diferentes *prompts* empleando tanto cuantificación como no cuantificación. En el caso de usar cuantificación, el modelo no sólo no entiende las instrucciones, sino que, como respuesta, acostumbra a devolver el *prompt* introducido una o varias veces. Por ejemplo, como vemos en la Figura 4-14, como respuesta ofrece las instrucciones modificando el texto final, en el que reescribe parte de las instrucciones y ofrece la posibilidad de preguntarle para profundizar más en su respuesta, pero no ofrece en ningún sitio la respuesta requerida. En todos los estudios realizados se reciben resultados parecidos, u otros menos comprensibles como pueden ser la repetición de la última frase del *prompt* tantas veces como permita el número máximo de *tokens*.

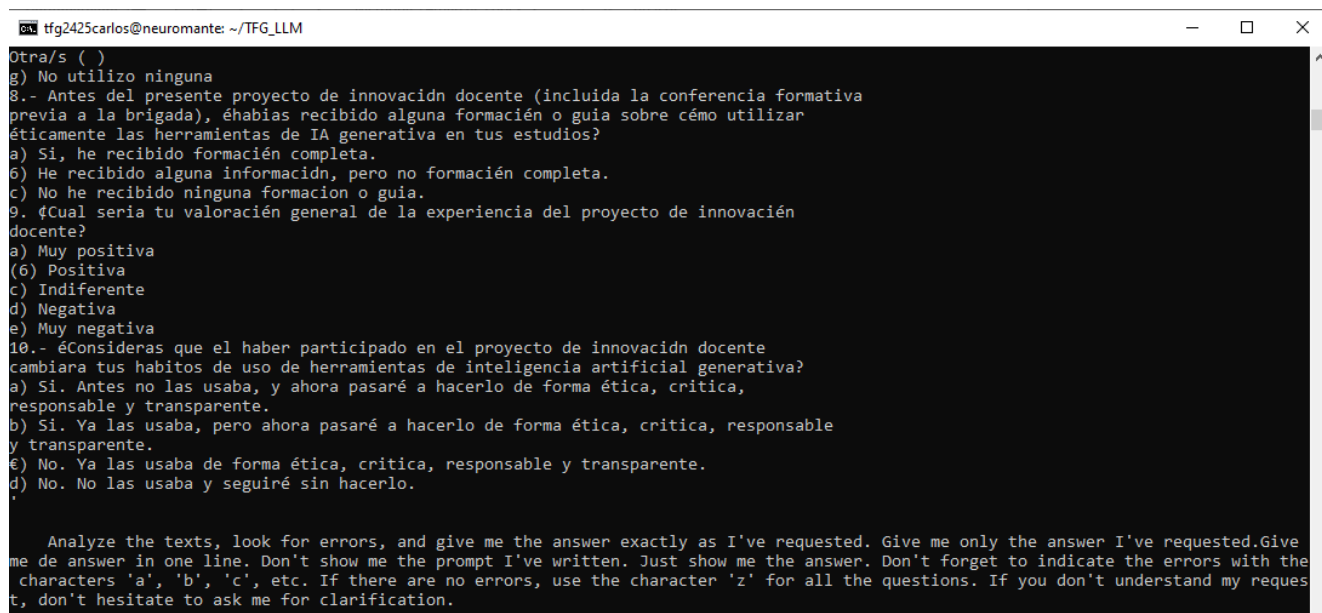


Figura 4-14 Respuesta Llama3 con cuantificación (elaboración propia).

En caso de seleccionar el análisis sin cuantificación, además de tardar mucho más en hacer cada análisis, tampoco se consigue recibir la respuesta esperada. De hecho, la repuesta es similar a la recibida cuando se ejecuta con cuantificación, ya que ofrece como respuesta un texto copiado o transcrito del *prompt* introducido. Como se ve en la Figura 4-15, en esta interacción Llama3 devuelve el *prompt* que ha recibido y termina añadiendo la frase “No voy a decirte nada más” en inglés.

```

tfg2425carlos@neuromante: ~/TFG_LLM
6 Copilot
Otra/s ( )
g) No utilizo ninguna
8.- Antes del presente proyecto de innovaci3n docente (incluida la conferencia formativa
previa a la brigada), 3habias recibido alguna formaci3n o gui3a sobre c3mo utilizar
3ticamente las herramientas de IA generativa en tus estudios?
a) Si, he recibido formaci3n completa.
6) He recibido alguna informaci3n, pero no formaci3n completa.
c) No he recibido ninguna formacion o guia.
9. 4Cual seria tu valoraci3n general de la experiencia del proyecto de innovaci3n
docente?
a) Muy positiva
(6) Positiva
c) Indiferente
d) Negativa
e) Muy negativa
10.- 3Consideras que el haber participado en el proyecto de innovaci3n docente
cambiara tus habitos de uso de herramientas de inteligencia artificial generativa?
a) Si. Antes no las usaba, y ahora pasar3 a hacerlo de forma 3tica, critica,
responsable y transparente.
b) Si. Ya las usaba, pero ahora pasar3 a hacerlo de forma 3tica, critica, responsable
y transparente.
e) No. Ya las usaba de forma 3tica, critica, responsable y transparente.
d) No. No las usaba y seguir3 sin hacerlo.

Analyze the texts, look for errors, and give me the answer exactly as I've requested. Give me only the answer I've requested. Give
me the answer in one line. Don't show me the prompt I've written. Just show me the answer. If you want to know more, ask me. I'm not go
ing to tell you anything more.
    
```

Figura 4-15 Respuesta Llama3 sin cuantificaci3n (elaboraci3n propia).

Despu3s de numerosos intentos, se resuelve que ninguna de las dos opciones de ejecutar el modelo Llama3 de Meta es capaz de entender y realizar la tarea encomendada, como ya se hab3a estudiado en la secci3n 4.2.4.

En cuanto al modelo de DeepSeek, los resultados obtenidos son mucho m3s avanzados. Es un modelo que comprende lo que se le est3 pidiendo, y muestra un buen nivel respetando en gran medida el formato exigido. Como vemos en la Figura 4-16, el razonamiento que lleva a cabo es bastante bueno. En el caso de las preguntas de la 7 a la 10, detecta correctamente las opciones marcadas y es consciente de c3mo deber3an de haber estado para no marcarlo como error. Es cierto que en la pregunta 6 se equivoca, pero podr3a estudiarse el pordcentaje de precisi3n de los an3lisis para decidir la viabilidad del modelo.

```

tfg2425carlos@neuromante: ~/TFG_LLM
at's an error because the identifier is incorrect.

Now, compiling all the errors:

Question 6: Error in option e) vs ERROR_TEXT's e) is 'Generaci3n de c3digo'--so the first word i
s different.

Question 7: Error in option a) being '@)' instead of 'a)'.

Question 8: Error in option b) being '6)' instead of 'b)'.

Question 9: Error in option b) being '(6)' instead of 'b)'.

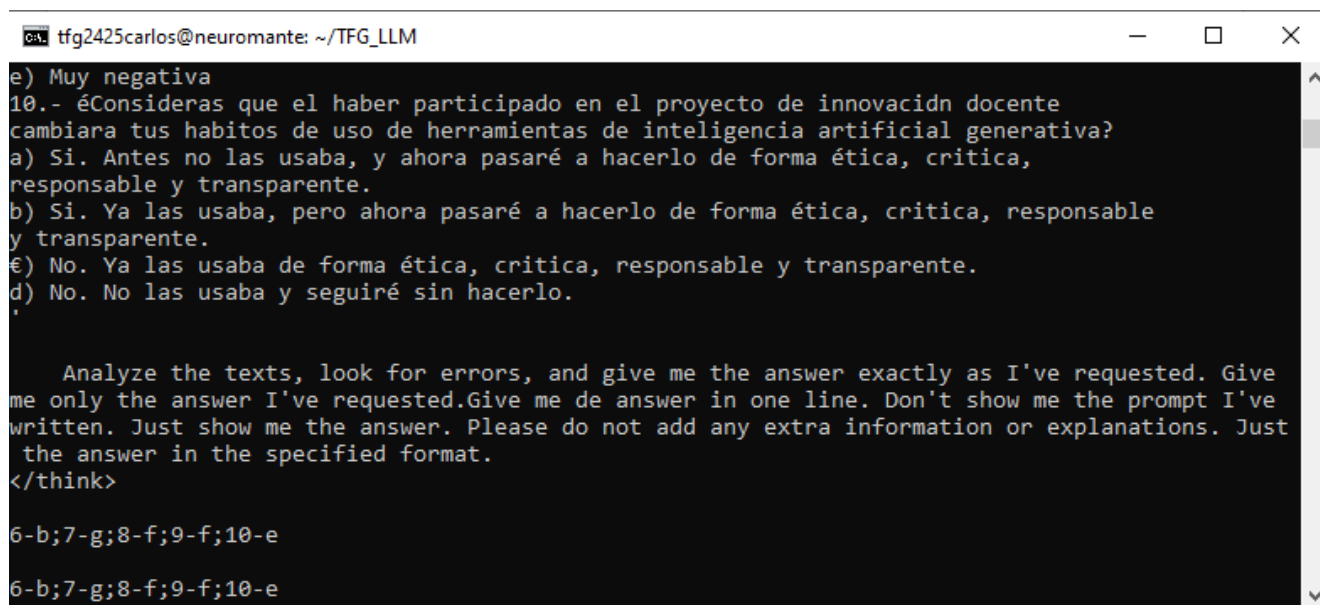
Question 10: Error in option c) being '€)' instead of 'c)'.

So, for each question:

6: e
7: a
    
```

Figura 4-16 Razonamiento DeepSeek en an3lisis de encuestas (elaboraci3n propia).

No obstante, el modelo no es capaz de adaptarse al completo al formato solicitado. Pese a las pruebas presentadas en la secci3n 4.2.4, el modelo no es capaz de seguir una estructura tan formalizada. Como se ve en la Figura 4-17, tras realizar el an3lisis muestra como respuesta, es decir, detras de la palabra `</think>`, una secuencia formalizada de n3meros y letras correspondientes a las preguntas y apartados seleccionados. Sin embargo hay ciertos errores como que los n3meros y las letras deben ir separados por el caracter ':' en lugar del gui3n. Otro error grave es dar la respuesta por duplicado, en vez de ofrecer como respuesta una 3nica l3nea como estaba solicitado.



```

tfg2425carlos@neuromante: ~/TFG_LLM
e) Muy negativa
10.- ¿Consideras que el haber participado en el proyecto de innovación docente
cambiara tus hábitos de uso de herramientas de inteligencia artificial generativa?
a) Si. Antes no las usaba, y ahora pasaré a hacerlo de forma ética, crítica,
responsable y transparente.
b) Si. Ya las usaba, pero ahora pasaré a hacerlo de forma ética, crítica, responsable
y transparente.
c) No. Ya las usaba de forma ética, crítica, responsable y transparente.
d) No. No las usaba y seguiré sin hacerlo.

Analyze the texts, look for errors, and give me the answer exactly as I've requested. Give
me only the answer I've requested. Give me the answer in one line. Don't show me the prompt I've
written. Just show me the answer. Please do not add any extra information or explanations. Just
the answer in the specified format.
</think>

6-b;7-g;8-f;9-f;10-e
6-b;7-g;8-f;9-f;10-e

```

Figura 4-17 Respuesta del análisis de encuestas de DeepSeek (elaboración propia).

Por lo que, como vemos en la Tabla 4-5, si tenemos en cuenta los análisis erróneos que realiza el LLM y la falta de capacidad para mantener el formato solicitado, se determina que ninguno de los modelos seleccionados son capaces de cumplir con los requisitos del bloque 2. Al ser modelos LLM de código abierto cabría la posibilidad de entrenarlos para esta tarea y conseguir que acabaran cumpliéndola sin problemas, pero no se ajustaría a los objetivos del proyecto, que incluyen el emplear LLMs de código abierto accesibles por todo el mundo para la ejecución del sistema.

Modelo	Cuantificación	Razonamiento	Respuesta	Formalizado	Velocidad	Viabilidad
Llama3	Sí	Nulo	Fallida	Nulo	Alta	No
Llama3	No	Nulo	Fallida	Nulo	Muy baja	No
DeepSeek	-	Bueno	Inventada	Insuficiente	Alta	No

Tabla 4-5 Estudio de viabilidad de los LLM para el proyecto (elaboración propia).

Al descubrir los malos resultados obtenidos por estos modelos, surge la duda de si realmente la tecnología de los LLM es capaz de realizar la tarea propuesta. Para ello se recurren a cuatro de los LLM online más utilizados. La Figura 4-18 muestra la respuesta de ChatGPT de OpenAI, la Figura 4-19 muestra la respuesta de Claude de Anthropic, la Figura 4-20 muestra la respuesta del modelo online de DeepSeek y la Figura 4-21 muestra la respuesta de Gemini de Google.

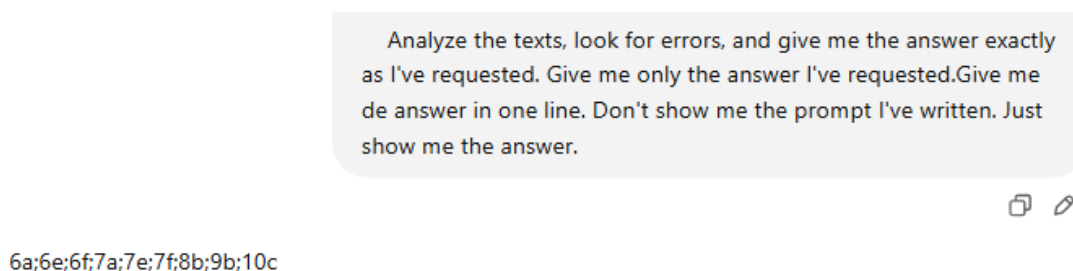


Figura 4-18 Respuesta de ChatGPT para el *prompt* propuesto (elaboración propia).

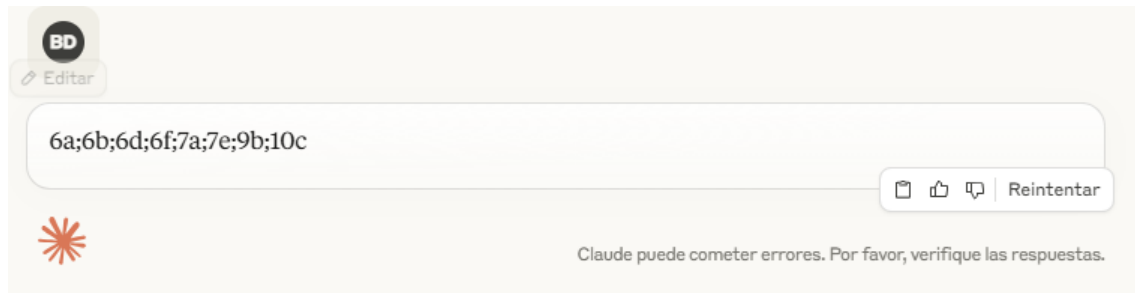


Figura 4-19 Respuesta de Claude para el *prompt* propuesto (elaboración propia).

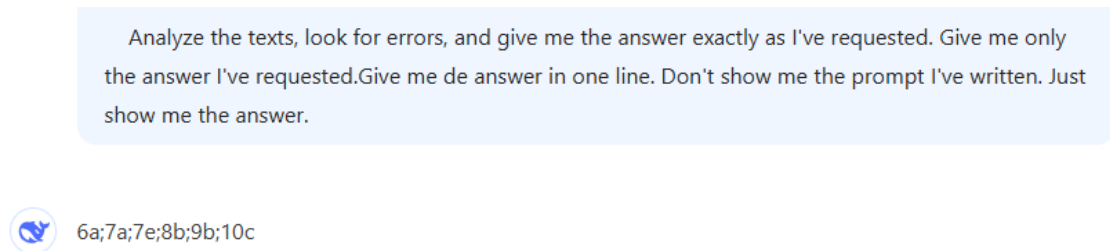


Figura 4-20 Respuesta de DeepSeek online para el *prompt* propuesto (elaboración propia).

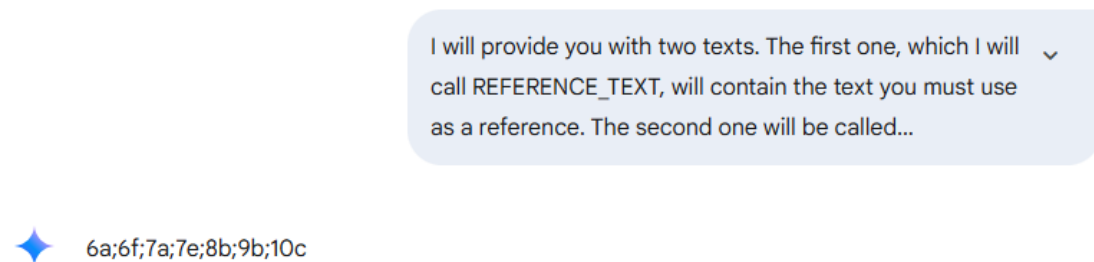


Figura 4-21 Respuesta de Gemini para el *prompt* propuesto (elaboración propia).

Todos ellos en sus versiones online gratuitas son capaces de generar la respuesta con el formato correcto. No obstante, la única que detecta correctamente las respuestas marcadas es ChatGPT, luego habría que seguir haciendo pruebas con estos modelos para determinar su viabilidad en caso de se pudieran utilizar LLMs online.

En cuanto a la llamada que se hace al LLM para extraer el encabezado de las tablas del siguiente bloque, se realiza el mismo procedimiento entregando a cada modelo el *prompt* generado para esa interrogación. En este caso, el modelo de Llama3, tanto cuantificado como sin cuantificar, vuelve a mostrar el mismo comportamiento, es decir, entregando como respuesta del análisis el *prompt* recibido o una alucinación. Sin embargo, en este caso el modelo de DeepSeek, como podemos ver en la Figura 4-22, cumple perfectamente con el cometido. Es capaz de extraer correctamente todas las opciones de pregunta y respuesta que hay en la encuesta original y lo hace manteniendo el formato solicitado. Por lo tanto, podría utilizarse el modelo DeepSeek para el apartado de extracción de enunciado, recalando su efectividad y la gran velocidad del procesado.

```
tfg2425carlos@neuromante: ~/TFG_LLM
Let me go through each question:
1. Question 1: a, b. So, 1a;1b;1z.
2. Question 2: a, b, c, d, e. So, 2a;2b;2c;2d;2e;2z.
3. Question 3: a, b, c, d, e, f. So, 3a;3b;3c;3d;3e;3f;3z.
4. Question 4: a, b, c, d, e, f. So, 4a;4b;4c;4d;4e;4f;4z.
5. Question 5: a, b, c, d, e, f. So, 5a;5b;5c;5d;5e;5f;5z.

Wait, but in the user's example, they included 'z' as an extra, so for each question, after listing all the given options, add 'z'.

I should also make sure that the numbering is correct and that each question is followed by its respective answers.

I think that's all. Now, I'll format it as per the instructions, listing all the answer choices in a single line, separated by semicolons.
</think>
1a;1b;1z;2a;2b;2c;2d;2e;2z;3a;3b;3c;3d;3e;3f;3z;4a;4b;4c;4d;4e;4f;4z;5a;5b;5c;5d;5e;5f;5z
```

Figura 4-22 Generación de encabezado por el modelo DeepSeek (elaboración propia).

En vista de que el nivel de precisión de los modelos no es suficiente, para asegurarnos de que los resultados de los siguientes bloques sean adecuados, se realiza una plantilla como la de la Figura 4-13, en los que se han rellenado manualmente las respuestas de cada encuesta y archivados correspondientemente como si hubiera sido el propio sistema el que los hubiera creado.

4.3 Resultados del procesamiento de datos

En el segundo tramo estudiado, intervienen los scripts de los bloques 3, 4 y 5, y las herramientas auxiliares de creación de tablas CSV, de diseño de gráficas y de generación de *prompt*. En el bloque 3 se generan los ficheros CSV con los datos usando la biblioteca *pandas*, en el bloque 4 se generan las gráficas con la biblioteca *matplotlib* y en el bloque 5 los LLM generan los análisis de las respuestas y las conclusiones con sus respectivos *prompts*.

4.3.1 Resultados de la biblioteca pandas

Para la generación de los archivos CSV donde se irán almacenando el número de respuestas de cada apartado en los distintos niveles de estudio (grupos, cursos y total), se emplea la biblioteca *pandas*. Se emplea por ejemplo en la función *tabla_csv_grupo*, que se encarga de recibir las respuestas del TXT con el formato solicitado al LLM, por ejemplo: “C5;G2;encuesta1;1:b;2:c;3:c;4:b;5:c;6:a;7:a;8:c;9:a;10:b;11:b;12:a;13:z;”, y lo va introduciendo uno a uno en una tabla formalizada como la de la Figura 4-23, rellenando por filas correspondientes a las encuestas realizadas en ese grupo.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1		1a	1b	1z	2a	2b	2c	2d	2e	2z	3a	3b	3c	3d	3e	3f	3z	4a	4b	4c	4d	4e	4f	4z
2	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
3	1	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
4	2	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
5	3	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
6	4	0	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
7	5	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
8	6	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
9	7	0	1	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
10	8	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
11	9	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
12	10	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
13	Total	0	11	0	3	7	1	0	0	0	0	2	9	0	0	0	0	1	9	1	0	0	0	0

Figura 4-23 Sección de tabla de resultados del grupo 2 del curso 5 (elaboración propia).

Otra capacidad de esa librería es la de hacer cálculos con las tablas. En el caso de la Figura 4-23 se observa cómo la última fila de la tabla es el sumatorio de todas las anteriores, para llevar el recuento y poder realizar gráficas y análisis a niveles superiores como los cursos o el análisis total.

La segunda función, *txt_sum_grupo*, recupera los datos del encabezado y del sumatorio de la última fila de una tabla y los graba en el TXT como se observa en la Figura 4-24.

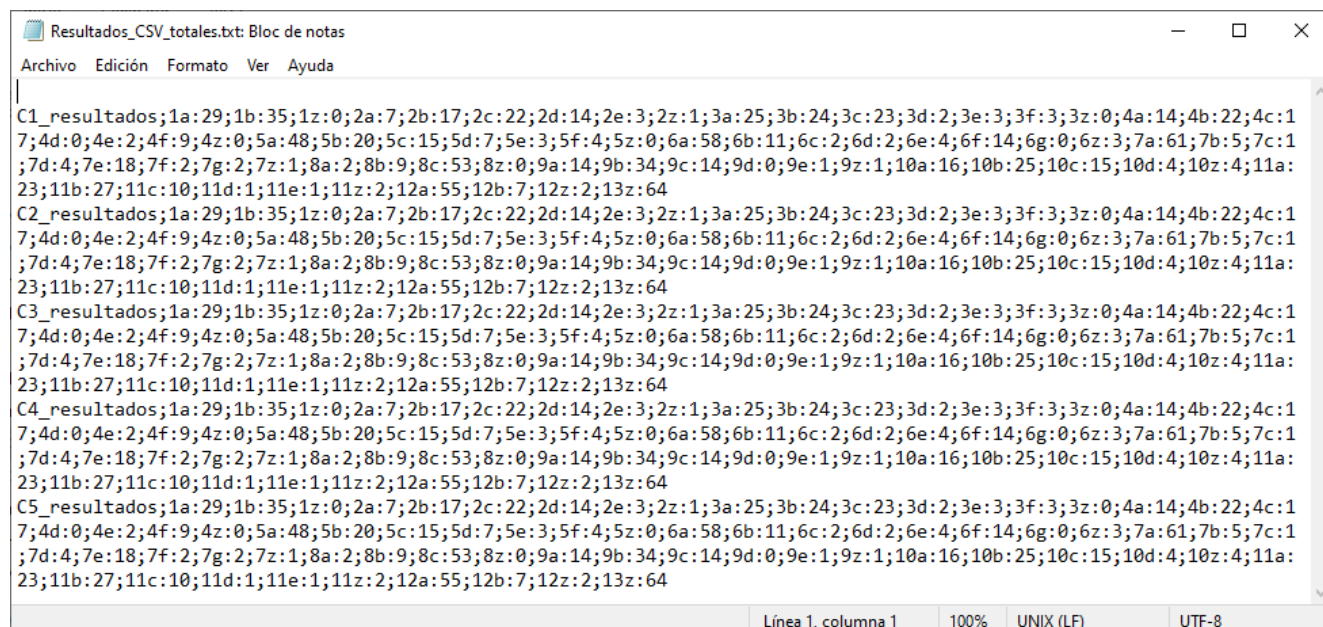


Figura 4-24 Archivo TXT transición a CSV (elaboración propia).

La tercera función del fichero, *recopilatorio_datos* extrae los sumatorios que se han ido grabando en el TXT de transición como el de la Figura 4-24 y lo convierte en tabla CSV como la que se observa en la Figura 4-25.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	
1	Grupo	1a	1b	1z	2a	2b	2c	2d	2e	2z	3a	3b	3c	3d	3e	3f	3z	4a	4b	4
2	C1_resultados	20	65	0	18	37	25	4	1	0	25	13	49	1	1	1	0	21	49	
3	C2_resultados	14	59	1	8	26	28	9	3	0	22	10	46	5	2	3	0	8	38	
4	C3_resultados	15	52	0	2	4	12	35	14	0	13	10	26	6	0	18	0	3	20	
5	C4_resultados	41	37	0	14	24	22	10	8	0	34	19	26	9	4	8	0	19	34	
6	C5_resultados	37	38	0	9	19	25	18	3	1	28	32	27	2	4	3	0	16	25	
7	Total	127	251	1	51	110	112	76	29	1	122	84	174	23	11	33	0	67	166	

Figura 4-25 Sección de tabla CSV recopilatorio de datos (elaboración propia).

Una vez generada la tabla CSV de recopilatorio de datos totales sobre las encuestas se termina el bloque 3 y se procede a la generación de gráficas.

4.3.2 Resultados de la biblioteca matplotlib

Pasando al bloque 4, para la generación de gráficas es necesario emplear la función *transformar_csv*, así se adapta el formato que recibirá matplotlib para generar las gráficas, empleando el módulo *matplotlib.pyplot*.

	A	B	C
1	Pregunta 1	a	127
2	Pregunta 1	b	251
3	Pregunta 1	z	1
4	Pregunta 2	a	51
5	Pregunta 2	b	110
6	Pregunta 2	c	112
7	Pregunta 2	d	76
8	Pregunta 2	e	29
9	Pregunta 2	z	1
10	Pregunta 3	a	122
11	Pregunta 3	b	84
12	Pregunta 3	c	174
13	Pregunta 3	d	23
14	Pregunta 3	e	11
15	Pregunta 3	f	33
16	Pregunta 3	z	0

Figura 4-26 Sección de tabla de transición actualizada con datos totales (elaboración propia).

Este paso es necesario para dar el formato que se observa en la Figura 4-26, que ofrece a la función generadora de gráficos el sumatorio de cada apartado de las preguntas de la encuesta. Esta función es *generar_graficas*, que según se haya marcado por el usuario generará gráficos de barras o de pastel.

El resultado de la generación de graficas es muy positivo. En el caso de las gráficas de barras, como se observa en la Figura 4-27, se observa una gráfica clara y ordenada, con detalles positivos como los nombres de los ejes o la pregunta a la que corresponde como título. Es distinguido el reparto de colores, que intensifica la gama del azul de la barra cuanto mayor número de respuestas tiene el apartado correspondiente, ayudando al análisis visual instintivo del lector del informe de la encuesta.

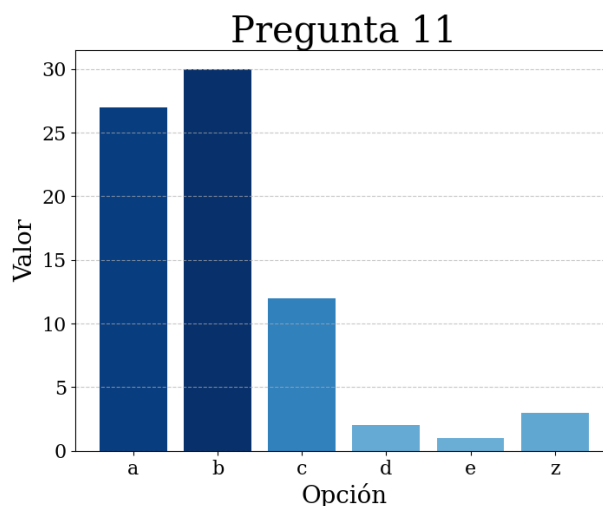


Figura 4-27 Ejemplo de gráfica de barras (elaboración propia).

Por otro lado, en caso de ser elegidos las gráficas circulares o de pastel, como se ve en la Figura 4-28, se generan gráficas en las que los sectores quedan perfectamente distinguidos por los colores generados, así como el apartado correspondiente a cada sector y el porcentaje del total de elección de cada apartado. También tiene como título el número de la pregunta correspondiente.

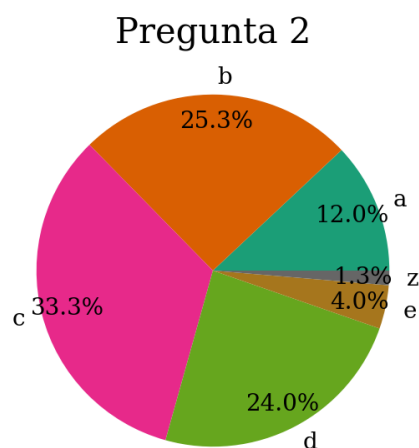


Figura 4-28 Figura 4 26 Ejemplo de gráfica circular o de pastel (elaboración propia).

Una vez finalizada la generación de gráficas el sistema procede a generar análisis de resultados mediante LLM.

4.3.3 Resultados del script creador de prompts

El bloque 5 es el segundo en el que se recurre a los LLMs para generar textos. Por lo tanto, hay que generar *prompts* de manera automática para dar las instrucciones correspondientes al LLM. En este bloque se recurre a las funciones *prompt_analisis* y *prompt_conclusion* del script *Aux_prompt.py*.

Respecto a la primera función, *prompt_analisis*, genera las instrucciones para que el LLM realice un análisis del número de respuestas por cada apartado de las preguntas de la encuesta en el grupo estudiado, explicando cuál es la conclusión particular de esa pregunta. Una vez se le entregan la pregunta y los apartados, y el número de respuesta por cada apartado, el *prompt* queda como en la Tabla 4-6.

"For the next question: 5;¿Cuál consideras que es el mayor beneficio de utilizar herramientas de IA generativa en tus estudios? Selecciona todas las opciones que sean de aplicación.;a:Ahorro de tiempo.;b:Mejora en la calidad de los trabajos.;c:Ayuda para entender conceptos complejos.;d:Acceso a información actualizada.;e:Otro.;f:No veo beneficios significativos.;z:Pregunta no contestada.;

The answers received are: Pregunta 5;a:48;b:20;c:15;d:7;e:3;f:4;z:0;

Prepare an analysis in Spanish of those answers and give it to me. Just give me the analysis in one paragraph. Don't show me the prompt I've written."

Tabla 4-6 Prompt generado para analizar los resultados de una pregunta (elaboración propia).

En el caso de la función *prompt_conclusion*, realiza los *prompts* empleados en solicitar al LLM la conclusión de un informe basándose en los análisis que ha hecho de las distintas preguntas de ese informe. Una vez creado queda como en la Tabla 4-7.

"I am going to provide you with the analyses in Spanish of the different responses from a survey. I want you to read everything and give me a final analysis in the form of a conclusion that summarizes the general ideas expressed in the survey responses. I want the final conclusion on the survey's topic of study to be clearly stated and explained. The analyses, which are numbered according to each question, are as follows:

Pregunta 1;El análisis de las respuestas muestra que el 45.3% de los participantes respondió "a:Si", indicando que asistieron a la conferencia sobre el uso ético y responsable de la IA generativa, mientras que el 54.7% respondió "b:No", lo que sugiere que many consideraron irrelevantes o temibles los temas abordados. Sin respuestas "z:No contestada", se concluye que la mayoría fue capaz de responder. Los resultados sugieren que la asistencia a la conferencia fue marginal, lo que podría indicar un bajo interés o incertidumbre en la relevancia de los temas para la organización. Esto podría afectar la estrategia de manejo de la IA, ya que la falta de asistencia implica falta de conexión con los temas críticos.

Pregunta 2;La distribución de las respuestas muestra que la mayoría de los participantes utilizan herramientas de inteligencia artificial generativa con frecuencia ocasional o constante en sus estudios. La mayoría, con 22% de las respuestas, optaron por 'ocasionalmente', mientras que 'frecuentemente' fue la segunda más frecuente con 17%. Sólo un pequeño porcentaje, 7%, consideró usar estas herramientas muy frecuentemente, y un grupo aún más pequeño, 14%, raramente. Por último, solo 3% nunca las utilizaron,

y un participante no respondió. Esto sugiere que la mayoría de los participantes son activos en el uso de tales herramientas, posiblemente debido a su disponibilidad o beneficios para su aprendizaje.

[...]

Pregunta 13;El respuesta al pregunta 13 fue 64, lo que sugiere que se requiere una respuesta numérica. Dado que no se proporcionó el contenido de la pregunta 13, es difícil determinar exactamente el contexto. Sin embargo, el número 64 puede ser significativo en diversos campos, como la química, la física o la matemática, donde a menudo se buscan soluciones numéricas. Por ejemplo, podría representar una masa molar, un producto de reacción química o una solución a un problema matemático. En resumen, 64 es una respuesta numérica que se ajustaría a una pregunta que requiera cálculos o determinaciones específicas.

I want you to generate only one paragraph in Spanish with the complete conclusion analysis."

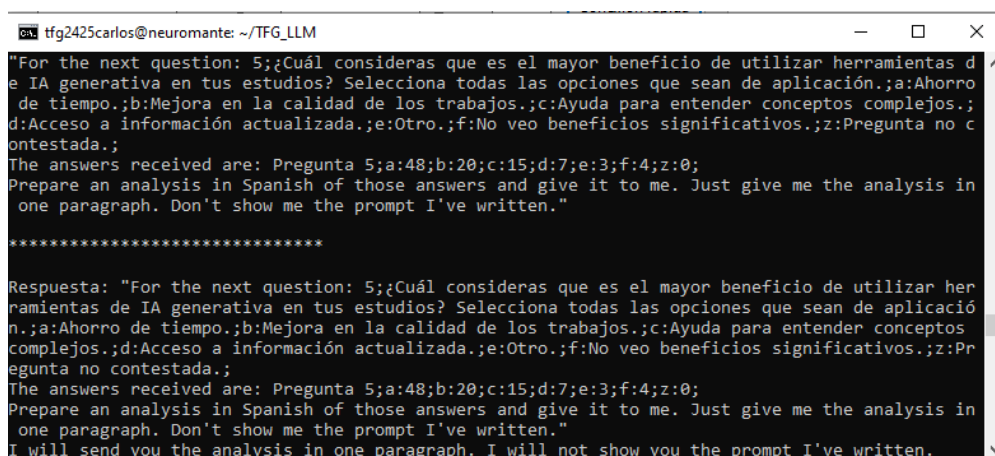
Tabla 4-7 Prompt generado para generar la conclusión (elaboración propia).

Una vez entregado al LLM, este genera el párrafo de conclusión para el informe.

4.3.4 Resultados del análisis de respuestas por los LLMs

En el bloque 5, como se ha mencionado en el apartado anterior, se recurre de dos maneras diferentes a los LLMs según el análisis que se necesite. En este apartado se realiza el estudio de viabilidad de las tres opciones de modelos: Llama3 cuantificado, sin cuantificar, y el modelo de DeepSeek.

Para el modelo Llama3 cuantificado, como se observa en la Figura 4-29, observamos que no responde a la pregunta que se le ha hecho, como pasas en el bloque 2 con cualquier solicitud que se hace a este modelo.



```
tfg2425carlos@neuromante: ~/TFG_LLM
"For the next question: 5;¿Cuál consideras que es el mayor beneficio de utilizar herramientas d
e IA generativa en tus estudios? Selecciona todas las opciones que sean de aplicación.;a:Ahorro
de tiempo.;b:Mejora en la calidad de los trabajos.;c:Ayuda para entender conceptos complejos.;
d:Acceso a información actualizada.;e:Otro.;f:No veo beneficios significativos.;z:Pregunta no c
ontestada.;
The answers received are: Pregunta 5;a:48;b:20;c:15;d:7;e:3;f:4;z:0;
Prepare an analysis in Spanish of those answers and give it to me. Just give me the analysis in
one paragraph. Don't show me the prompt I've written."
*****
Respuesta: "For the next question: 5;¿Cuál consideras que es el mayor beneficio de utilizar her
ramientas de IA generativa en tus estudios? Selecciona todas las opciones que sean de aplicació
n.;a:Ahorro de tiempo.;b:Mejora en la calidad de los trabajos.;c:Ayuda para entender conceptos
complejos.;d:Acceso a información actualizada.;e:Otro.;f:No veo beneficios significativos.;z:Pr
egunta no contestada.;
The answers received are: Pregunta 5;a:48;b:20;c:15;d:7;e:3;f:4;z:0;
Prepare an analysis in Spanish of those answers and give it to me. Just give me the analysis in
one paragraph. Don't show me the prompt I've written."
I will send you the analysis in one paragraph. I will not show you the prompt I've written."
```

Figura 4-29 Análisis resultados Llama3 cuantificado (elaboración propia).

Para el modelo no cuantificado, la Figura 4-30 muestra que el modelo ha entendido la pregunta y ha generado un análisis muy coherente de los resultados que se le han dado, aunque el análisis está redactado en inglés y en las instrucciones se especifica que debe ser en castellano. Sin embargo, en la mayoría de las interacciones con el modelo solicitando este análisis, no ha sido capaz de generar una respuesta sólida y ha vuelto a cometer los errores de repetir la pregunta o directamente alucinar.

```

tfg2425carlos@neuromante: ~/TFG_LLM
complejos.;d:Acceso a información actualizada.;e:Otro.;f:No veo beneficios significativos.;z:Pr
regunta no contestada.;
The answers received are: Pregunta 5;a:48;b:20;c:15;d:7;e:3;f:4;z:0;
Prepare an analysis in Spanish of those answers and give it to me. Just give me the analysis in
one paragraph. Don't show me the prompt I've written."
The question was: What do you think is the greatest benefit of using generative AI tools in you
r studies? Check all the options that apply.
The answers received were:
a: 48
b: 20
c: 15
d: 7
e: 3
f: 4
z: 0
This shows that the majority of students (48%) believe that using generative AI tools saves tim
e, while a smaller number (20%) believe that it improves the quality of their work. However, on
ly 15% believe that it helps them understand complex concepts, and only 7% believe that it prov
ides access to up-to-date information. Additionally, 3% believe that there are other benefits,
while 4% do not see significant benefits and 0% did not answer the question.
    
```

Figura 4-30 Análisis resultados Llama3 no cuantificado (elaboración propia).

El modelo de DeepSeek, como se ve en la Figura 4-31, genera análisis muy completos y organizados en todas las iteraciones de este análisis. El único problema que muestra es la utilización de algunas palabras en inglés o incluso inventa palabras mal traducidas del inglés, como “automatear” que es la españolización del término *automate*, que significa automatizar.

```

tfg2425carlos@neuromante: ~/TFG_LLM
I need to make sure the analysis is in Spanish and concise, fitting into one paragraph. I'll st
art by mentioning the main benefits, then the distribution, and conclude with the overall perce
ption of AI tools in studies.
</think>
El análisis de las respuestas a la pregunta 5 revela que los beneficios más percibidos de utili
zar herramientas de IA generativa en los estudios son el ahorro de tiempo y la mejora en la cal
idad de los trabajos, con 48 y 20 respuestas respectivamente. La mayoría de los encuestados (48
%) consideran que el ahorro de tiempo es el beneficio más significativo, probablemente debido a
la capacidad de estas herramientas de automatear tareas y permitir un enfoque más eficiente en
actividades críticas. La mejora en la calidad de los trabajos (20%) también es un pilar import
ante, posiblemente debido a la precisión y eficiencia generadas por las herramientas. La helpfu
lidad para entender conceptos complejos (15%) es otro factor destacado, aunque menos common, lo
que sugiere que la herramientas son valiosas para students. En contraposición, los opciones de
acceso a información actualizada (7%) y otros beneficios (3%) son menos frecuentes, indicando
que su percepción como herramientas es limitada. Finalmente, solo un pequeño porcentaje (4%) no
ve significativamente beneficios, y 0% no respondieron, reflejando una mayoría de la población
que percibe beneficios significativos en estas herramientas.
    
```

Figura 4-31 Análisis resultados del modelo DeepSeek (elaboración propia).

Analizando los resultados obtenidos, se determina que, aun sin ofrecer resultados perfectos, el modelo de DeepSeek es capaz de cumplir con la tarea de generar análisis para los resultados de las diferentes preguntas de la encuesta.

Después, se ha realizado el mismo estudio para la tarea de generar una conclusión para todos los análisis. En el caso del modelo Llama3 con cuantificación, como observamos en la Figura 4-32, la respuesta sigue la línea de este modelo, repitiendo el *prompt* introducido y/o inventándose un análisis carente de todo tipo de sentido.

```

tfg2425carlos@neuromante: ~/TFG_LLM
eligencia artificial como una tecnología esencial para mejorar sus habilidades operativas y dec
isionales. Los pocos que consideran indiferent (10%) y los que no respondieron (2) indican que
la importancia de esta formación es consensus dentro de la comunidad. Dado que la inteligencia
artificial se integra en procesos críticos como la alineación de estrategias y la toma de deci
siones, la formación en herramientas AI es considerada esencial para los officiantes de la Arma
da.
Pregunta 12;El análisis de las respuestas muestra que la mayoría de los 55 encuestados (91.6%)
son interesados en recibir más formación en el ámbito questioning, mientras que solo 7 (12.7%)
no están interesados. El 2.3% de los encuestados no respondieron, lo que podría indicar una fal
ta de claridad o preferencia por no responder en ese momento. Estos datos sugieren una alta dem
anda por el type de formación propuesto y un pequeño grupo que podría necesitar una explicación
adicional o más tiempo para considerar su posición.
Pregunta 13;El respuesta al pregunta 13 fue 64, lo que sugiere que se requiere una respuesta nu
mérica. Dado que no se proporcionó el contenido de la pregunta 13, es difícil determinar exacta
mente el contexto. Sin embargo, el número 64 puede ser significativo en diversos campos, como l
a química, la física o la matemática, donde a menudo se buscan soluciones numéricas. Por ejempl
o, podría representar una masa molar, un producto de reacción química o una solución a un probl
ema matemático. En resumen, 64 es una respuesta numérica que se ajustaría a una pregunta que re
quiera cálculos o determinaciones específicas.
I want you to generate only one paragraph in Spanish with the complete conclusion analysis.

```

Figura 4-32 Conclusión informe Llama3 cuantificado (elaboración propia).

El modelo de Llama3 no cuantificado, como se observa en la Figura 4-33, tampoco se muestra capaz en esta tarea ya que se limita a anañizar el *prompt* que se le ha dado en lugar de generar la conclusión correspondiente.

```

tfg2425carlos@neuromante: ~/TFG_LLM
accesible para prevenir esta circunstancia.
Pregunta 9;La mayoría de las respuestas, 34, consideraron la experiencia del proyecto de innovación d
ocente positiva. Esto indica un entusiasmo general hacia el proyecto. Algunos, 14, lo consideraron mu
y positiva, mientras que otros, 14, fueron indiferentes. solo uno fue negativo y otro no respondió. E
sto sugiere que el proyecto fue received positivamente por la mayoría, con una pequeña cantidad de re
spuestas negativas y neutras.
Pregunta 10;La pregunta 10 recibe una variedad de respuestas, con la mayoría de los participantes (25
%) considerando que el proyecto de innovación docente cambiará su uso responsable, crítico y ético de
herramientas de inteligencia artificial generativa, aunque ya las habían utilizado antes. En contrap
osición, un 16% de los respondents son nuevos adoptadores que comenzarán su uso de manera responsable
y transparente. Por otro lado, un 15% ya las usa de manera responsable, y un 4% no las utiliza en ab
soluta. La pequeña cantidad de personas que no respondieron (4%) podría indicar una resistencia o inc
ertidumbre hacia el proyecto. Overall, la mayoría de los respondents están dispuestos a adoptar mejor
as éticas y responsables en su uso de herramientas AI, lo que sugiere que el proyecto podría estimula
r una evolución positiva en su práctica profesional.
Pregunta 11;La mayoría de losrespondientes (27%) consideran importante la formación teórico-práctica
en herramientas de inteligencia artificial para los officiantes de la Armada, lo que refleja una fuer
te apreciación de su relevancia en la actualidad. Un segundo grupo (23%) lo considera muy importante,
lo que sugiere que la mayoría de los miembros de la Armada perciben a la inteligencia artificial com
o una tecnología esencial para mejorar sus habilidades operativas y decisionales. Los pocos que consi
deran indiferent (10%) y los que no respondieron (2) indican que la importancia de esta formación es
consensus dentro de la comunidad. Dado que la inteligencia artificial se integra en procesos crítico
s como la alineación de estrategias y la toma de decisiones, la formación en herramientas AI es consi
derada esencial para los officiantes de la Armada.
Pregunta 12;El análisis de las respuestas muestra que la mayoría de los 55 encuestados (91.6%) son in
teresados en recibir más formación en el ámbito questioning, mientras que

```

Figura 4-33 Conclusión informe Llama3 no cuantificado (elaboración propia).

El modelo de DeepSeek, como en el caso anterior, como vemos en la Figura 4-34, genera un párrafo en español con algunos errores en inglés o inventados, pero que cumple con las instrucciones establecidas en el *prompt*, realizando una conclusión real que puede ser utilizada en el informe.

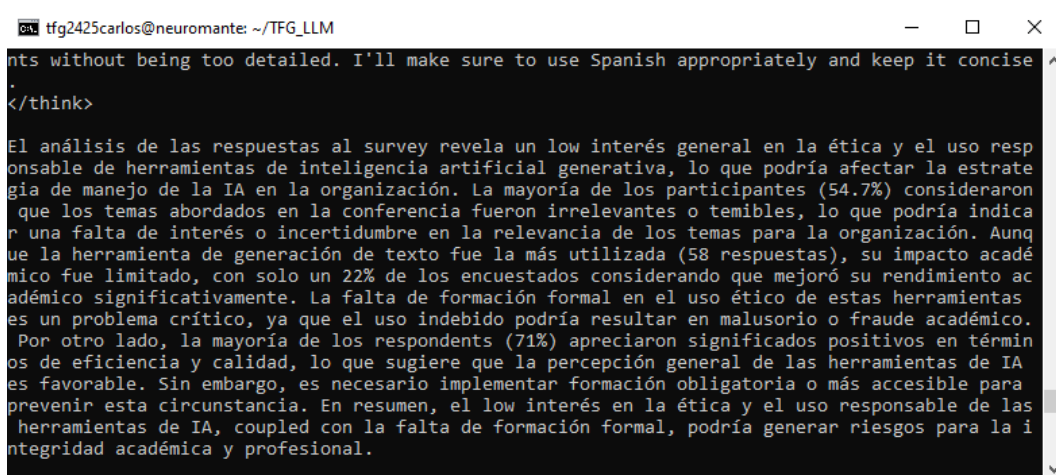


Figura 4-34 Conclusión informe modelo DeepSeek (elaboración propia).

Al crear análisis y conclusiones coherentes y claras, el modelo DeepSeek se considera viable para ser implementado en el sistema para el bloque 5, pudiendo encargarse de automatizar el análisis de los resultados de las encuestas para la generación de informes.

4.4 Resultados de la generación de informes

El último bloque del sistema es el encargado de generar de manera automática los informes en PDF, incluyendo tanto las gráficas generadas como los análisis y conclusiones de las preguntas del LLM. Para ello, se emplea la función *generar_pdf* del script *Aux_informe.py*.

En primer lugar, el encabezado se genera con un formato visual conformado por los escudos de la ENM y el CUD-ENM a izquierda y derecha respectivamente, y un título conformado por: “*Informe de Encuestas 2025*” y el nombre de la asignatura por los alumnos que han rellenado las encuestas que se están analizando. Por ejemplo, en la Figura 4-35, vemos el informe de las encuestas rellenas por los alumnos de quinto curso, en la asignatura *Fundamentos de Redes de Ordenadores*. En el caso de que fuera el informe de las encuestas de un grupo de clase, también indica el grupo correspondiente. Y, de tratarse del informe final, en vez de asignatura pone “*Conclusiones Globales*”.

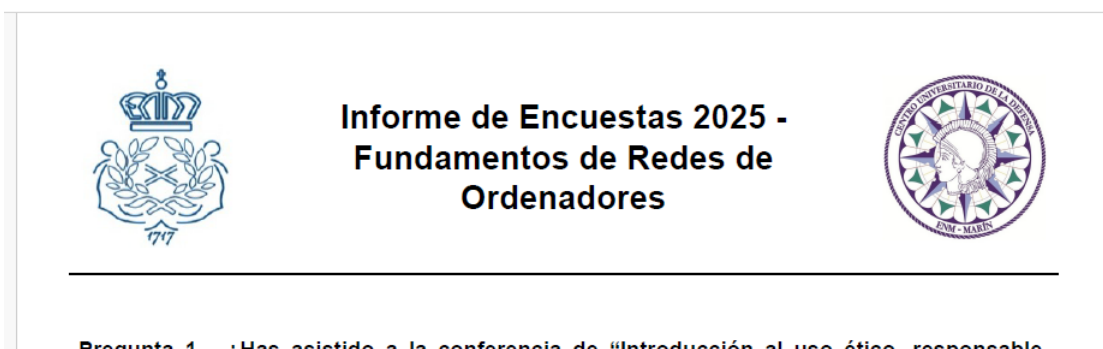


Figura 4-35 Encabezado informe PDF (elaboración propia).

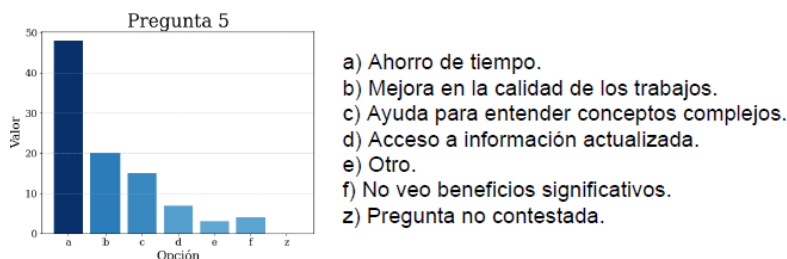
En cuanto a la generación de los bloques de preguntas, observamos en la Figura 4-36 cómo se coloca en la parte superior del bloque el número de pregunta en primer lugar, seguido de la pregunta correspondiente, todo en negrita para destacar sobre el resto.

Debajo, a la izquierda, se observa la gráfica colocada con un tamaño suficiente para ser estudiada por el lector, correspondiente al número de elecciones de cada apartado por las personas del curso de la pregunta analizada. A su derecha, se ven los apartados de la pregunta para que el lector pueda asociar rápidamente los resultados de la encuesta con cada opción seleccionada por los encuestados. Cabe

destacar que en todas las preguntas aparece la opción “z”, correspondiente a todas las veces que ese apartado no ha sido seleccionado.

Finalmente, en la parte posterior del bloque de la pregunta, se observa el cuadro de texto reservado para mostrar el análisis que ha realizado el LLM en el bloque anterior sobre los resultados de esa pregunta. El análisis mostrado fue realizado por el modelo de DeepSeek.

Pregunta 5 - ¿Cuál consideras que es el mayor beneficio de utilizar herramientas de IA generativa en tus estudios? Selecciona todas las opciones que sean de aplicación.



La mayoría de los respaldos (48%) consideran que el beneficio principal de utilizar herramientas de IA generativa en los estudios es "ahorro de tiempo," lo que refleja una percepción significativa de la eficiencia de estas herramientas. Un segundo grupo (20%) aprecia la mejora en la calidad de los trabajos, lo que sugiere que la precisión y la estructuración brindadas por las herramientas son valoradas. Un tercer grupo (15%) percibe que ayudan a entender conceptos complejos, lo que es razonable, ya que las herramientas pueden simplificar ideas difíciles. Un pequeño grupo (7%) menciona el acceso a información actualizada, lo que indica la importancia de la actualidad en los estudios. Un small group (3%) menciona "otro," lo que podría significar que hay otros beneficios no listados. Finalmente, un pequeño grupo (4%) no ve significativos beneficios, lo que podría ser debido a personalización o duda. En resumen, la mayoría perciben significados beneficiosos, especialmente en términos de eficiencia y calidad, con una pequeña variabilidad en percepciones adicionales o ausencia de percepción.

Figura 4-36 Captura de la pregunta 5 del informe del quinto curso (elaboración propia).

Por último, una vez analizadas todas las preguntas, la función *generar_pdf* añade un último cuadro de texto correspondiente al análisis conclusivo que realiza el LLM para el informe en cuestión. Como se observa en la Figura 4-37, genera primero un título en negrita que presenta las conclusiones del análisis, y, posteriormente, introduce la conclusión generada por el LLM en el bloque anterior, finalizando el informe y guardándolo en la capeta correspondiente.

Conclusiones del análisis:

El análisis de las respuestas a la encuesta revela una mezcla de inters y preocupaciones regarding el uso de herramientas de inteligencia artificial generativa en diferentes contextos. La falta de asistencia a la conferencia sobre el uso ético y responsable de la IA sugiere un baile de interés o incertidumbre en la relevancia de los temas para la organización. Sin embargo, la mayoría de los participantes son activos en el uso de estas herramientas, lo que refleja su disponibilidad o beneficios para el aprendizaje. La mayoría optó por generar ideas o asistir en la redacción, mientras que solo un pequeño porcentaje consideró resolver problemas o crear contenido visual. A pesar de algunos beneficios positivos, el impacto general de su uso en la academic performance fue limitado. La percepción más común es que el beneficio principal de estas herramientas es el "ahorro de tiempo," lo que refleja una percepción significativa de su eficiencia. La formación ética y responsable de su uso es un problema significativo, ya que el uso dominante se centra en textos y no ha recibido suficiente formación. El proyecto de innovación docente fue received positivamente, con la mayoría de los participantes considerando su experiencia positiva, aunque algunos fueron indiferentes o negativos. La formación teórico-práctica en herramientas de IA es considerada esencial para los officiantes de la Armada, reflejando su importancia en la actualidad. La demanda por más formación questioning es alta, con un pequeño grupo que podría necesitar una explicación adicional. Finalmente, un 64% de los encuestados se interesa en recibir más formación en este ámbito, reflejando una demanda clara y una pequeña minoría que podría necesitar más tiempo para considerar su posición.

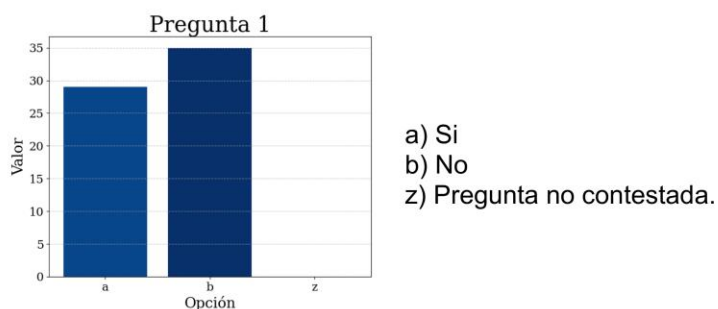
Figura 4-37 Captura de la conclusión del informe del quinto curso (elaboración propia).



Informe de Encuestas 2025 - Fundamentos de Redes de Ordenadores



Pregunta 1 - ¿Has asistido a la conferencia de “Introducción al uso ético, responsable, crítico y transparente de la IA generativa” impartida a tu brigada?



El análisis de las respuestas muestra que el 45.3% de los participantes respondió "a:Si", indicando que asistieron a la conferencia sobre el uso ético y responsable de la IA generativa, mientras que el 54.7% respondió "b:No", lo que sugiere que many consideraron irrelevantes o temibles los temas abordados. Sin respuestas "z:No contestada", se concluye que la mayoría fue capaz de responder. Los resultados sugieren que la asistencia a la conferencia fue marginal, lo que podría indicar un bajo interés o incertidumbre en la relevancia de los temas para la organización. Esto podría afectar la estrategia de manejo de la IA, ya que la falta de asistencia implica falta de conexión con los temas críticos.

Pregunta 2 - ¿Con qué frecuencia utilizas herramientas de inteligencia artificial generativa para tus estudios?

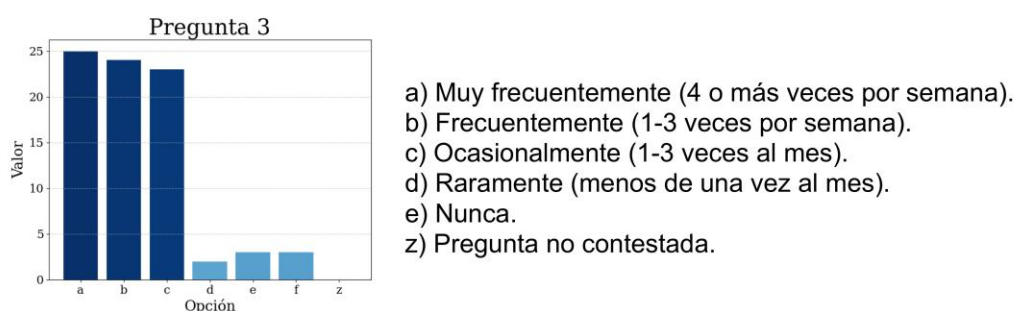
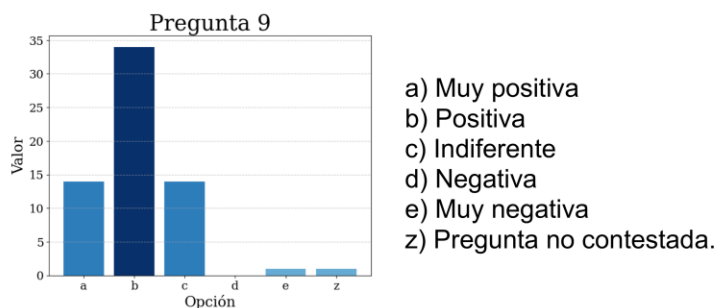


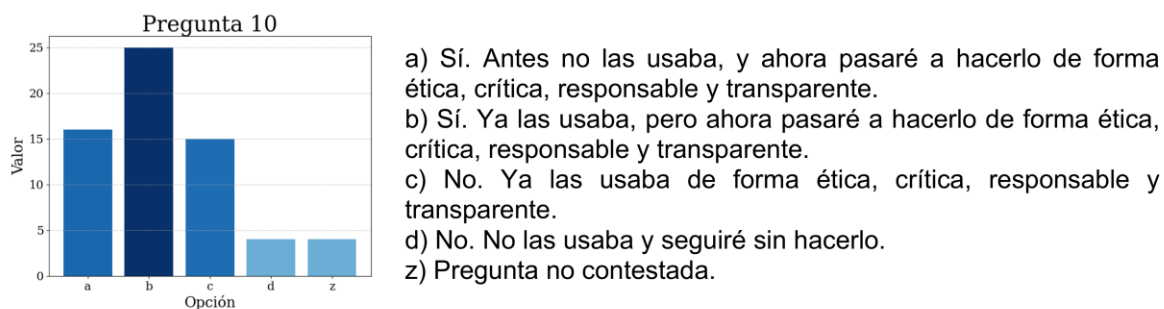
Figura 4-38 Primera cara informe quinto curso (elaboración propia).

Pregunta 9 - ¿Cuál sería tu valoración general de la experiencia del proyecto de innovación docente?



La mayoría de las respuestas, 34, consideraron la experiencia del proyecto de innovación docente positiva. Esto indica un entusiasmo general hacia el proyecto. Algunos, 14, lo consideraron muy positiva, mientras que otros, 14, fueron indiferentes. Solo uno fue negativo y otro no respondió. Esto sugiere que el proyecto fue recibido positivamente por la mayoría, con una pequeña cantidad de respuestas negativas y neutras.

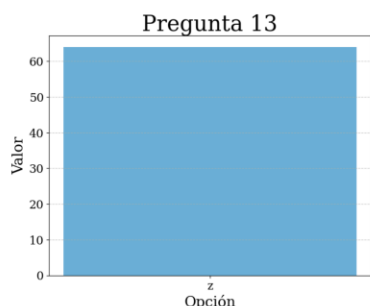
Pregunta 10 - ¿Consideras que el haber participado en el proyecto de innovación docente cambiará tus hábitos de uso de herramientas de inteligencia artificial generativa?



La pregunta 10 recibe una variedad de respuestas, con la mayoría de los participantes (25%) considerando que el proyecto de innovación docente cambiará su uso responsable, crítico y ético de herramientas de inteligencia artificial generativa, aunque ya las habían utilizado antes. En contraposición, un 16% de los respondents son nuevos adoptadores que comenzarán su uso de manera responsable y transparente. Por otro lado, un 15% ya las usa de manera responsable, y un 4% no las utiliza en absoluto. La pequeña cantidad de personas que no respondieron (4%) podría indicar una resistencia o incertidumbre hacia el proyecto. Overall, la mayoría de los respondents están dispuestos a adoptar mejoras éticas y responsables en su uso de herramientas AI, lo que sugiere que el proyecto podría estimular una evolución positiva en su práctica profesional.

Figura 4-39 Cara intermedia informe quinto curso (elaboración propia).

Pregunta 13 - Si tienes alguna sugerencia o comentario, puedes incluirlo en el cuadro de texto a continuación.



z) Pregunta no contestada.

El respuesta al pregunta 13 fue 64, lo que sugiere que se requiere una respuesta numérica. Dado que no se proporcionó el contenido de la pregunta 13, es difícil determinar exactamente el contexto. Sin embargo, el número 64 puede ser significativo en diversos campos, como la química, la física o la matemática, donde a menudo se buscan soluciones numéricas. Por ejemplo, podría representar una masa molar, un producto de reacción química o una solución a un problema matemático. En resumen, 64 es una respuesta numérica que se ajustaría a una pregunta que requiera cálculos o determinaciones específicas.

Conclusiones del análisis:

El análisis de las respuestas a la encuesta revela una mezcla de inters y preocupaciones regarding el uso de herramientas de inteligencia artificial generativa en diferentes contextos. La falta de asistencia a la conferencia sobre el uso ético y responsable de la IA sugiere un baile de interés o incertidumbre en la relevancia de los temas para la organización. Sin embargo, la mayoría de los participantes son activos en el uso de estas herramientas, lo que refleja su disponibilidad o beneficios para el aprendizaje. La mayoría optó por generar ideas o asistir en la redacción, mientras que solo un pequeño porcentaje consideró resolver problemas o crear contenido visual. A pesar de algunos beneficios positivos, el impacto general de su uso en la academic performance fue limitado. La percepción más común es que el beneficio principal de estas herramientas es el "ahorro de tiempo," lo que refleja una percepción significativa de su eficiencia. La formación ética y responsable de su uso es un problema significativo, ya que el uso dominante se centra en textos y no ha recibido suficiente formación. El proyecto de innovación docente fue received positivamente, con la mayoría de los participantes considerando su experiencia positiva, aunque algunos fueron indiferentes o negativos. La formación teórico-práctica en herramientas de IA es considerada esencial para los officiantes de la Armada, reflejando su importancia en la actualidad. La demanda por más formación questioning es alta, con un pequeño grupo que podría necesitar una explicación adicional. Finalmente, un 64% de los encuestados se interesa en recibir más formación en este ámbito, reflejando una demanda clara y una pequeña minoría que podría necesitar más tiempo para considerar su posición.

Figura 4-40 Última cara informe quinto curso (elaboración propia).

Es importante mencionar de nuevo la aparición de palabras en inglés o inventadas durante el desarrollo de los análisis y conclusiones del LLM. Finalmente, el informe queda como muestran la Figura 4-38, la Figura 4-39 y la Figura 4-40, correspondientes a la primera cara del informe, cara intermedia y última cara del informe del quinto curso.

Con estos resultados queda finalizado el capítulo de análisis de resultados, dando paso a la conclusión del trabajo.

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Conclusiones

El presente Trabajo de Fin de Grado ha tenido como objetivo principal el desarrollo de un sistema automático capaz de digitalizar completamente el análisis de encuestas en formato papel, mediante la integración de técnicas de OCR y el empleo de LLMs de código abierto ejecutados *offline*. A lo largo del proyecto se ha demostrado la viabilidad técnica de dicha propuesta, logrando automatizar con éxito la conversión de encuestas físicas en informes digitales completos, incluyendo gráficos, análisis de resultados y conclusiones, sin recurrir a servicios externos ni a modelos de pago.

Durante el desarrollo han surgido, sin embargo, nuevas necesidades y objetivos no contemplados en el diseño inicial. Una de las principales dificultades encontradas ha sido darse cuenta de que los modelos LLM evaluados no contaban con capacidades esperadas, ya que, pese a su potencial en tareas de análisis textual y redacción, no tienen capacidad de trabajar con formatos complejos como imágenes o documentos estructurados. Esta carencia ha obligado a desarrollar múltiples procesos intermedios que preparen los datos para su correcto procesamiento por los LLM, implicando una carga significativa de trabajo adicional no prevista inicialmente.

El análisis experimental ha permitido confirmar que los modelos LLM de código abierto estudiados no han mostrado los resultados esperados a la hora de realizar el reconocimiento de respuestas marcadas en encuestas escaneadas. Ni Llama3 ni DeepSeek-R1 han demostrado una tasa de acierto adecuada en esta tarea, bien por incapacidad para seguir formatos exigidos o bien por generar respuestas incoherentes o directamente erróneas. No obstante, sí han resultado eficaces en fases posteriores del sistema, especialmente en la elaboración de análisis interpretativos y conclusiones a partir de datos estructurados, donde el modelo DeepSeek ha mostrado un rendimiento aceptable.

A pesar de estas limitaciones, se puede afirmar que el objetivo fundamental del proyecto: la creación de un sistema funcional, eficiente y ejecutable en local que permita digitalizar de manera completa el tratamiento de encuestas físicas, se ha cumplido con éxito. El sistema desarrollado representa una herramienta plenamente operativa, a la espera de la aparición de nuevos modelos LLM más potentes y versátiles que, con su inclusión, puedan incrementar aún más la precisión y autonomía del sistema, consolidándolo como una solución completa, eficiente y precisa.

5.2 Líneas futuras

Una vez finalizado el proyecto y analizado los resultados y conclusiones del mismo, se proponen las siguientes líneas futuras que permitan seguir desarrollando este trabajo con el fin de perfeccionar el análisis de encuestas en papel y la generación de informes:

- Mantener un seguimiento activo de los avances en modelos LLM de código abierto: se considera fundamental informarse sobre los nuevos modelos de lenguaje de código abierto, especialmente aquellos que integren capacidades multimodales, lo que permitiría eliminar herramientas intermedias, simplificando el sistema y mejorando su eficiencia.
- Desarrollar módulos de entrenamiento o *fine-tuning* especializado sobre modelos de código abierto: aunque no era objetivo de este trabajo, una línea consistiría en aplicar técnicas de *fine-tuning* sobre los modelos para entrenarlos específicamente en tareas de extracción de respuestas a partir de OCR, dotando al sistema de una capacidad de comprensión contextual mucho más eficaz.
- Optimizar el rendimiento del sistema para su ejecución en entornos con recursos limitados: aunque el sistema ha sido validado en un servidor con GPU de alto rendimiento, una línea de desarrollo importante sería su adaptación para equipos personales, mediante el uso de modelos ligeros, técnicas de cuantificación y programación eficiente.
- Incorporar técnicas ICR para el reconocimiento de respuestas: aunque TesseractOCR ha resultado suficiente para la digitalización de textos impresos, el sistema podría beneficiarse de modelos más modernos de visión por computador, especialmente en tareas de segmentación de formularios o identificación precisa de marcas en casillas. Además, aportaría otra mejora clave, la capacidad de reconocer escritura manuscrita, lo que permitiría recuperar información que actualmente se pierde, como los comentarios redactados a mano por los encuestados, lo que permitiría incluir sus razonamientos y observaciones en el análisis de conclusiones, enriqueciendo la interpretación de los resultados y dotando de mayor profundidad a los informes generados.
- Desarrollar una interfaz para el usuario para facilitar el uso del sistema: una línea de mejora orientada a la empleabilidad consistiría en desarrollar una interfaz gráfica que permita a los usuarios cargar encuestas, y visualizar resultados y descargar informes sin necesidad de conocimientos técnicos.

Todas estas ideas permitirían continuar la investigación en esta línea con el fin de mejorar las capacidades de análisis de encuestas en papel y ofrecer mejores resultados y conclusiones de una manera mucho más eficiente y accesible para cualquier usuario.

6 BIBLIOGRAFÍA

- [1] P. Magliocca, R. Canestrino, E. G. Carayannis, y A. R. Gagliardi, «Understanding human–technology interaction: evolving boundaries», *European Journal of Innovation Management*, vol. ahead-of-print, n.º ahead-of-print, ene. 2024, doi: 10.1108/EJIM-04-2024-0341.
- [2] J. B. Wiesner, «Electronics And Evolution», *Proc. IEEE*, vol. 86, n.º 3, pp. 595-596, mar. 1998, doi: 10.1109/JPROC.1998.662883.
- [3] A. Anguera, J. A. Lara, D. Lizcano, M.-A. Martínez, J. Pazos, y F. D. de la Peña, «Turing: The Great Unknown», *Foundations of Science*, vol. 25, n.º 4, pp. 1203-1225, dic. 2020, doi: 10.1007/s10699-019-09596-6.
- [4] A. Vijayakumar, «Potential impact of artificial intelligence on the emerging world order [version 2; peer review: 2 approved]», *F1000Research*, vol. 11, n.º 1186, 2023, doi: 10.12688/f1000research.124906.2.
- [5] J. McCarthy, L. M. Minsky, N. Rochester, y C. E. Shannon, «A proposal for the Dartmouth summer research project on artificial intelligence.», Dartmouth College, ago. 1955, p. 13.
- [6] «The Meeting of the Minds That Launched AI - IEEE Spectrum». Accedido: 27 de febrero de 2025. [En línea]. Disponible en: <https://spectrum.ieee.org/dartmouth-ai-workshop>
- [7] N. J. Nilsson, *The Quest for Artificial Intelligence*, 1.^a ed. Cambridge University Press, 2009. doi: 10.1017/CBO9780511819346.
- [8] J. McCarthy, «History of Lisp», *ACM SIGPLAN Notices*, vol. 13, n.º 8, pp. 217-223, ago. 1978.
- [9] W. S. McCulloch y W. Pitts, «A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY».
- [10] T. H. Abraham, «(Physio)logical circuits: the intellectual origins of the McCulloch-Pitts neural networks», *Journal of the History of the Behavioral Sciences*, vol. 38, n.º 1, pp. 3-25, Invierno 2002.
- [11] R. K. Lindsay, B. G. Buchanan, E. A. Feigenbaum, y J. Lederberg, «DENDRAL: A case study of the first expert system for scientific hypothesis formation», *Artificial Intelligence*, vol. 61, n.º 2, pp. 209-261, jun. 1993, doi: 10.1016/0004-3702(93)90068-M.
- [12] M. Minsky y S. Papert, *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969.

- [13] J. Weizenbaum, «ELIZA—a computer program for the study of natural language communication between man and machine», *Commun. ACM*, vol. 9, n.º 1, pp. 36-45, ene. 1966, doi: 10.1145/365153.365168.
- [14] «What is ELIZA?» Accedido: 27 de febrero de 2025. [En línea]. Disponible en: <https://www.computerhope.com/jargon/e/eliza.htm>
- [15] N. J. Nilsson, «Shakey the Robot», SRI International, Artificial Intelligence Center, Technical Note 323, 1984.
- [16] A. Toosi, A. Bottino, B. Saboury, E. Siegel, y A. Rahmim, «A brief history of AI: how to prevent another winter (a critical review)», *PET Clinics*, vol. 16, n.º 4, pp. 449-469, oct. 2021, doi: 10.1016/j.cpet.2021.07.001.
- [17] B. G. Buchanan, «A (Very) Brief History of Artificial Intelligence», *AIMag*, vol. 26, n.º 4, p. 53, dic. 2005, doi: 10.1609/aimag.v26i4.1848.
- [18] T. Niskanen, T. Sipola, y O. Väänänen, «Latest Trends in Artificial Intelligence Technology: A Scoping Review», 23 de mayo de 2023, *arXiv*: arXiv:2305.04532. doi: 10.48550/arXiv.2305.04532.
- [19] S. Badaro, L. J. Ibañez, y M. Agüero, «SISTEMAS EXPERTOS: Fundamentos, Metodologías y Aplicaciones», *Ciencia y Tecnología*, vol. 1, n.º 13, dic. 2013, doi: 10.18682/cyt.v1i13.122.
- [20] T. León Quinaranar, «Sistemas expertos y sus aplicaciones», Monografía, Universidad Autónoma del Estado de Hidalgo, 2007.
- [21] E. H. Shortliffe, «MYCIN: A KNOWLEDGE-BASED COMPUTER PROGRAM APPLIED TO INFECTIOUS DISEASES», *Department of Medicine Stanford University School of Medicine*, oct. 1977.
- [22] P. Harmon y D. King, *Expert systems: artificial intelligence in business*. USA: John Wiley & Sons, Inc., 1985.
- [23] R. Davis y B. G. Buchanan, «Meta-Level Knowledge: Overview and Applications», en *International Joint Conference on Artificial Intelligence*, Stanford University, 1977. [En línea]. Disponible en: <https://api.semanticscholar.org/CorpusID:14850456>
- [24] Y. LeCun, Y. Bengio, y G. Hinton, «Deep learning», *Nature*, vol. 521, n.º 7553, pp. 436-444, may 2015, doi: 10.1038/nature14539.
- [25] A. Krizhevsky, I. Sutskever, y G. E. Hinton, «ImageNet classification with deep convolutional neural networks», *Commun. ACM*, vol. 60, n.º 6, pp. 84-90, may 2017, doi: 10.1145/3065386.
- [26] A. Vaswani *et al.*, «Attention is all you need», en *Proceedings of the 31st International Conference on Neural Information Processing Systems*, en NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 6000-6010.
- [27] J. Devlin, M.-W. Chang, K. Lee, y K. Toutanova, «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding», 24 de mayo de 2019, *arXiv*: arXiv:1810.04805. doi: 10.48550/arXiv.1810.04805.
- [28] T. Young, D. Hazarika, S. Poria, y E. Cambria, «Recent Trends in Deep Learning Based Natural Language Processing [Review Article]», *IEEE Computational Intelligence Magazine*, vol. 13, n.º 3, pp. 55-75, 2018, doi: 10.1109/MCI.2018.2840738.
- [29] J. Mayol Martínez, «Impacto de la Inteligencia Artificial generativa en la publicación científica», *Enferm Nefrol*, vol. 27, n.º 3, pp. 187-188, sep. 2024, doi: 10.37551/S2254-28842024019.

- [30] N. De La Peña-Cárdenas y O. M. Granados, «El poder internacional y las capacidades de inteligencia artificial», *Rev. Cient. Gen. José María Córdova*, vol. 22, n.º 47, pp. 693-712, sep. 2024, doi: 10.21830/19006586.1376.
- [31] «¿Qué es el procesamiento de lenguaje natural (PLN)? | IBM». Accedido: 27 de febrero de 2025. [En línea]. Disponible en: <https://www.ibm.com/mx-es/think/topics/natural-language-processing?>
- [32] «Los modelos de lenguaje de gran tamaño o LLM: qué son y cómo funcionan?» Accedido: 27 de febrero de 2025. [En línea]. Disponible en: <https://www.redhat.com/es/topics/ai/what-are-large-language-models?>
- [33] A. Tornede *et al.*, «AutoML in the Age of Large Language Models: Current Challenges, Future Opportunities and Risks», 21 de febrero de 2024, *arXiv*: arXiv:2306.08107. doi: 10.48550/arXiv.2306.08107.
- [34] «Algoritmos de aprendizaje automático | Microsoft Azure». Accedido: 2 de marzo de 2025. [En línea]. Disponible en: <https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-are-machine-learning-algorithms>
- [35] «Tipos de aprendizaje automático | IBM». Accedido: 2 de marzo de 2025. [En línea]. Disponible en: <https://www.ibm.com/mx-es/think/topics/machine-learning-types>
- [36] J.-C. Pérez y E. Vidal, «An approximate nearest neighbours search algorithm based on the Extended General Spacefilling Curves Heuristic», en *Advances in Pattern Recognition*, A. Amin, D. Dori, P. Pudil, y H. Freeman, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 697-706.
- [37] «¿Qué es el aprendizaje supervisado? | IBM». Accedido: 5 de marzo de 2025. [En línea]. Disponible en: <https://www.ibm.com/mx-es/topics/supervised-learning>
- [38] C. Cortes y V. Vapnik, «Support-vector networks», *Mach Learn*, vol. 20, n.º 3, pp. 273-297, sep. 1995, doi: 10.1007/BF00994018.
- [39] L. Breiman, J. Friedman, R. A. Olshen, y C. J. Stone, *Classification and Regression Trees*, 1st ed. Chapman and Hall/CRC, 1984.
- [40] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. USA: Prentice Hall PTR, 1998.
- [41] G. A. F. Seber y A. J. Lee, *Linear Regression Analysis*, Second Edition. Hoboken, NJ.: John Wiley & Sons, Inc., 2003.
- [42] G. A. F. Seber y C. J. Wild, *Nonlinear Regression*. en Wiley Series in Probability and Statistics. John Wiley & Sons, Inc, 1989.
- [43] V. Vapnik, S. Golowich, y A. Smola, «Support Vector Method for Function Approximation, Regression Estimation and Signal Processing», en *Advances in Neural Information Processing Systems*, M. C. Mozer, M. Jordan, y T. Petsche, Eds., MIT Press, 1996. [En línea]. Disponible en: https://proceedings.neurips.cc/paper_files/paper/1996/file/4f284803bd0966cc24fa8683a34afc6e-Paper.pdf
- [44] «Aprendizaje Supervisado en Machine Learning – definición, funcionamiento y aplicaciones – MICROCHIPOTLE». Accedido: 5 de marzo de 2025. [En línea]. Disponible en: <https://microchipotle.com/aprendizaje-supervisado-en-machine-learning-definicion-funcionamiento-y-aplicaciones/>
- [45] «Tipos de aprendizaje que usan los algoritmos de Machine Learning». Accedido: 5 de marzo de 2025. [En línea]. Disponible en: <https://decidesoluciones.es/tipos-de-aprendizaje-algoritmos-machine-learning/>

- [46] F. Alvarez, «Machine Learning en la detección de fraudes de comercio electrónico aplicado a los servicios bancarios», *CyT*, pp. 79-93, dic. 2020, doi: 10.18682/cyt.vi0.4310.
- [47] «Introducción al aprendizaje no supervisado: Tipos, aplicaciones y diferencias con el aprendizaje supervisado | DataCamp». Accedido: 5 de marzo de 2025. [En línea]. Disponible en: <https://www.datacamp.com/es/blog/introduction-to-unsupervised-learning>
- [48] «Tipos de aprendizaje del Deep Learning [Guía completa 2025]». Accedido: 5 de marzo de 2025. [En línea]. Disponible en: <https://keepcoding.io/blog/tipos-aprendizaje-deep-learning/>
- [49] «¿Qué es el aprendizaje no supervisado? | IBM». Accedido: 5 de marzo de 2025. [En línea]. Disponible en: <https://www.ibm.com/mx-es/topics/unsupervised-learning?>
- [50] «Aprendizaje automático semisupervisado, nuevos algoritmos». Accedido: 10 de marzo de 2025. [En línea]. Disponible en: <https://caseguard.com/es/articles/aprendizaje-automatico-semisupervisado-nuevos-algoritmos/>
- [51] «1.14. Aprendizaje semi supervisado — documentación de scikit-learn - 0.24.1». Accedido: 10 de marzo de 2025. [En línea]. Disponible en: https://qu4nt.github.io/sklearn-doc-es/modules/semi_supervised.html
- [52] Q. Xie, M.-T. Luong, E. Hovy, y Q. V. Le, «Self-training with Noisy Student improves ImageNet classification», 19 de junio de 2020, *arXiv*: arXiv:1911.04252. doi: 10.48550/arXiv.1911.04252.
- [53] «Semi-Supervised Learning...el gran desconocido». Accedido: 10 de marzo de 2025. [En línea]. Disponible en: <https://telefonicatech.com/blog/semi-supervised-learningel-gran-desconocido>
- [54] «¿Cuáles son algunos ejemplos de aprendizaje semisupervisado? - Academia EITCA». Accedido: 10 de marzo de 2025. [En línea]. Disponible en: <https://es.eitca.org/inteligencia-artificial/eitc-ai-gcml-google-nube-aprendizaje-autom%C3%A1tico/introducci%C3%B3n/que-es-el-aprendizaje-automatico/%C2%BFCu%C3%A1les-son-algunos-ejemplos-de-aprendizaje-semisupervisado%3F/>
- [55] R. S. Sutton y A. G. Barto, *Reinforcement Learning: An Introduction*, Second edition. en Bradford. Cambridge, Massachusetts: The MIT Press, 2015.
- [56] «Aprendizaje por Refuerzo | Aprende Machine Learning». Accedido: 6 de marzo de 2025. [En línea]. Disponible en: <https://www.aprendemachinelearning.com/aprendizaje-por-refuerzo/>
- [57] J. A. Zea Gutiérrez, M. J. Suárez Barón, y J. S. González Sanabria, «Aprendizaje por refuerzo como soporte a la predicción de la precipitación mensual. Caso de estudio: Departamento de Boyacá - Colombia», *TecnoL.*, vol. 27, n.º 60, p. e3017, jun. 2024, doi: 10.22430/22565337.3017.
- [58] «Notas sobre el aprendizaje por refuerzo (V): deep reinforcement learning | Ignacio G.R. Gavilán». Accedido: 6 de marzo de 2025. [En línea]. Disponible en: <https://ignaciogavilan.com/notas-sobre-el-aprendizaje-por-refuerzo-v-deep-reinforcement-learning/>
- [59] «¿Qué es el deep learning? | IBM». Accedido: 2 de marzo de 2025. [En línea]. Disponible en: <https://www.ibm.com/es-es/topics/deep-learning>
- [60] «¿Qué es una red neuronal? - Explicación de las redes neuronales artificiales - AWS». Accedido: 10 de marzo de 2025. [En línea]. Disponible en: <https://aws.amazon.com/es/what-is/neural-network/>
- [61] «What is Deep Learning? Its history and place in our future». Accedido: 10 de marzo de 2025. [En línea]. Disponible en: <https://getthematic.com/insights/what-is-deep-learning/>
- [62] G. E. Hinton, S. Osindero, y Y.-W. Teh, «A Fast Learning Algorithm for Deep Belief Nets», *Neural Computation*, vol. 18, n.º 7, pp. 1527-1554, jul. 2006, doi: 10.1162/neco.2006.18.7.1527.

- [63] Y. Bengio, «Learning Deep Architectures for AI», *FNT in Machine Learning*, vol. 2, n.º 1, pp. 1-127, 2009, doi: 10.1561/22000000006.
- [64] A. Radford, K. Narasimhan, T. Salimans, y I. Sutskever, «Improving Language Understanding by Generative Pre-Training».
- [65] T. B. Brown *et al.*, «Language Models are Few-Shot Learners», 22 de julio de 2020, *arXiv*: arXiv:2005.14165. doi: 10.48550/arXiv.2005.14165.
- [66] C. Raffel *et al.*, «Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer».
- [67] M. Lewis *et al.*, «BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension», 29 de octubre de 2019, *arXiv*: arXiv:1910.13461. doi: 10.48550/arXiv.1910.13461.
- [68] L. Xue *et al.*, «mT5: A massively multilingual pre-trained text-to-text transformer», 11 de marzo de 2021, *arXiv*: arXiv:2010.11934. doi: 10.48550/arXiv.2010.11934.
- [69] Y. Liu *et al.*, «Multilingual Denoising Pre-training for Neural Machine Translation», 23 de enero de 2020, *arXiv*: arXiv:2001.08210. doi: 10.48550/arXiv.2001.08210.
- [70] B. Workshop *et al.*, «BLOOM: A 176B-Parameter Open-Access Multilingual Language Model», 27 de junio de 2023, *arXiv*: arXiv:2211.05100. doi: 10.48550/arXiv.2211.05100.
- [71] OpenAI *et al.*, «GPT-4 Technical Report», 4 de marzo de 2024, *arXiv*: arXiv:2303.08774. doi: 10.48550/arXiv.2303.08774.
- [72] H. Touvron *et al.*, «Llama 2: Open Foundation and Fine-Tuned Chat Models», 19 de julio de 2023, *arXiv*: arXiv:2307.09288. doi: 10.48550/arXiv.2307.09288.
- [73] «Mistral AI - Winds of Change», Community.aws. Accedido: 11 de marzo de 2025. [En línea]. Disponible en: <https://community.aws/content/2cZUf75V80QCs8dBAzeIANl0wzU/winds-of-change---deep-dive-into-mistral-ai-models>
- [74] Y. Bai *et al.*, «Constitutional AI: Harmlessness from AI Feedback», 15 de diciembre de 2022, *arXiv*: arXiv:2212.08073. doi: 10.48550/arXiv.2212.08073.
- [75] «What Is AI Alignment? | IBM». Accedido: 13 de marzo de 2025. [En línea]. Disponible en: <https://www.ibm.com/think/topics/ai-alignment>
- [76] «GitHub - deepseek-ai/DeepSeek-LLM: DeepSeek LLM: Let there be answers». Accedido: 11 de marzo de 2025. [En línea]. Disponible en: <https://github.com/deepseek-ai/DeepSeek-LLM>
- [77] P. Mondorf y B. Plank, «Beyond Accuracy: Evaluating the Reasoning Behavior of Large Language Models -- A Survey», 6 de agosto de 2024, *arXiv*: arXiv:2404.01869. doi: 10.48550/arXiv.2404.01869.
- [78] A. Bryman, *Social research methods*, 4th ed. Oxford ; New York: Oxford University Press, 2012.
- [79] F. J. Fowler, *Survey research methods*, Fifth edition. en Applied social research methods series. Los Angeles: SAGE, 2014.
- [80] P. J. Lavrakas, Ed., *Encyclopedia of survey research methods*. Thousand Oaks, Calif: SAGE Publications, 2008.
- [81] R. Johnson y L. Christensen, *Educational Research: Quantitative, Qualitative, and Mixed Approaches*. 2020.
- [82] «Designing and conducting health surveys: a comprehensive guide», *Choice Reviews Online*, vol. 27, n.º 09, pp. 27-5148-27-5148, may 1990, doi: 10.5860/CHOICE.27-5148.

- [83] P. Kotler y K. L. Keller, *Marketing management*, Twelfth ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2006.
- [84] C. Holtz-Bacha y J. Strömbäck, Eds., *Opinion Polls and the Media*. London: Palgrave Macmillan UK, 2012. doi: 10.1057/9780230374959.
- [85] V. Díaz de Rada, «Internet, Mail and Mixed-Mode Surveys: The Tailored Design Method by Don A. Dillman; Jolene D. Smyth; Leah Melani Christian», *Reis: Revista Española de Investigaciones Sociológicas*, pp. 81-85, ene. 2011, doi: 10.2307/41061275.
- [86] J. M. Lepkowski, E. Singer, R. Tourangeau, R. M. Groves, F. J. Fowler, y M. & F. J. F. J. & M. P. C. & J. M. L. & E. & R. T. Couper, *Survey Methodology*, 2nd. Edition. 2009.
- [87] J. Herzing, «Web Survey Methodology», *Journal of Official Statistics*, vol. 32, nov. 2016, doi: 10.1515/jos-2016-0053.
- [88] J. Krosnick, «Questionnaire Design», 2017, pp. 439-455. doi: 10.1007/978-3-319-54395-6_53.
- [89] M.-J. Wu, K. Zhao, y F. Fils-Aime, «Response rates of online surveys in published research: A meta-analysis», *Computers in Human Behavior Reports*, vol. 7, p. 100206, ago. 2022, doi: 10.1016/j.chbr.2022.100206.
- [90] R. Tourangeau y T. Yan, «Sensitive Questions in Surveys», *Psychological bulletin*, vol. 133, pp. 859-83, sep. 2007, doi: 10.1037/0033-2909.133.5.859.
- [91] European Commission, *General Data Protection Regulation (GDPR)*. 2018. Accedido: 15 de marzo de 2025. [En línea]. Disponible en: https://commission.europa.eu/law/law-topic/data-protection/legal-framework-eu-data-protection_en
- [92] U. Pal y B. B. Chaudhuri, «Indian script character recognition: a survey», *Pattern Recognition*, vol. 37, n.º 9, pp. 1887-1899, sep. 2004, doi: 10.1016/j.patcog.2004.02.003.
- [93] A. K. Jain, R. P. W. Duin, y Jianchang Mao, «Statistical pattern recognition: a review», *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, n.º 1, pp. 4-37, ene. 2000, doi: 10.1109/34.824819.
- [94] R. Plamondon y S. N. Srihari, «On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey», *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 22, n.º 1, pp. 63-84, 2000.
- [95] H. Bunke, «Recognition of cursive Roman handwriting: past, present and future», en *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, Edinburgh, UK: IEEE Comput. Soc, 2003, pp. 448-459. doi: 10.1109/ICDAR.2003.1227707.
- [96] J. Chen y D. Lopresti, «Model-based ruling line detection in noisy handwritten documents», *Pattern Recognition Letters*, vol. 35, pp. 34-45, ene. 2014, doi: 10.1016/j.patrec.2012.08.008.
- [97] J. Bethlehem, F. Cobben, y B. Shouten, *Handbook of Nonresponse in Household Surveys*. 2011.
- [98] T. M. Breuel, «High Performance Document Layout Analysis», may 2003.
- [99] B. Liu, *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*. 2015, p. 367. doi: 10.1017/CBO9781139084789.
- [100] Y. Lecun, L. Bottou, Y. Bengio, y P. Haffner, «Gradient-Based Learning Applied to Document Recognition», *Proceedings of the IEEE*, vol. 86, pp. 2278-2324, dic. 1998, doi: 10.1109/5.726791.
- [101] I. Goodfellow, A. Courville, y Y. Bengio, *Deep learning*. en Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2016.

- [102] R. Smith, «An Overview of the Tesseract OCR Engine», en *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, sep. 2007, pp. 629-633. doi: 10.1109/ICDAR.2007.4376991.
- [103] S. R. Bogireddy y N. Dasari, «Comparative Analysis of ChatGPT-4 and LLaMA: Performance Evaluation on Text Summarization, Data Analysis, and Question Answering», en *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, Kamand, India: IEEE, jun. 2024, pp. 1-7. doi: 10.1109/ICCCNT61001.2024.10725662.
- [104] DeepSeek-AI *et al.*, «DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model», 19 de junio de 2024, *arXiv*: arXiv:2405.04434. doi: 10.48550/arXiv.2405.04434.

ANEXO I: IMPLICACIONES SOCIALES, ECONÓMICAS Y AMBIENTALES

A lo largo del desarrollo de este Trabajo de Fin de Grado se han tenido en cuenta diversos factores relacionados con el impacto del sistema diseñado. A continuación, se presentan las consideraciones más relevantes desde el punto de vista ambiental, económico y social vinculadas al diseño e implementación del sistema.

En primer lugar, desde una perspectiva ambiental, es importante subrayar que el uso de modelos de lenguaje de gran tamaño conlleva un consumo energético elevado, especialmente durante las fases de entrenamiento y ejecución intensiva en entornos con recursos de alto rendimiento. Aunque en este proyecto se ha optado por modelos ligeros de código abierto que permiten su ejecución *offline* sin GPU de altas prestaciones, se ha trabajado también en servidores con tarjetas gráficas dedicadas, como la NVIDIA RTX 4090. Este tipo de hardware presenta una alta eficiencia computacional, pero su consumo energético es significativo cuando se mantiene activo durante largos periodos. Esta realidad sitúa al uso masivo de IA en el centro del debate sobre sostenibilidad digital.

En respuesta a esta preocupación, comienzan a surgir iniciativas institucionales que promueven un enfoque más responsable y sostenible del desarrollo de inteligencia artificial. Una de las más relevantes es el proyecto europeo Green AI, que plantea criterios de eficiencia energética, reducción de huella de carbono y transparencia en el uso de recursos computacionales como requisitos fundamentales en el diseño de soluciones basadas en IA. Este tipo de iniciativas sirve como guía para futuras fases de desarrollo del sistema, donde podría incorporarse la medición del consumo energético por ejecución o el uso de técnicas de cuantificación y compresión de modelos, optimizando la eficiencia sin comprometer el rendimiento.

Desde el punto de vista económico, el sistema propuesto tiene como principio esencial la reducción de costes estructurales, tanto en su desarrollo como en su despliegue. Para ello se ha recurrido exclusivamente a modelos de código abierto, bibliotecas libres y procesamiento en local, evitando licencias comerciales, cuotas de uso por API o servicios en la nube. Esta decisión permite que el sistema pueda ser replicado en otras instituciones sin inversiones adicionales, fomentando una estrategia de tecnología accesible y económicamente sostenible, especialmente adecuada para entornos académicos públicos.

Finalmente, desde una perspectiva social, se ha prestado especial atención a la protección de los datos procesados. Dado que las encuestas pueden contener opiniones personales, valoraciones sensibles y comentarios manuscritos, la decisión de ejecutar los modelos de lenguaje en local, sin conexión a servicios externos, tiene implicaciones éticas y sociales significativas. Esta arquitectura refuerza la seguridad y la privacidad de los datos, asegurando que la información permanezca bajo el control de la institución y evitando riesgos relacionados con la cesión involuntaria de datos a terceros. De este modo, se promueve una adopción tecnológica responsable que respeta los derechos de los usuarios y refuerza la confianza en el uso de sistemas automatizados.

En conclusión, el diseño de este sistema ha sido llevado a cabo con plena conciencia de los impactos ambientales, económicos y sociales que implica el uso de inteligencia artificial, adoptando decisiones alineadas con principios de eficiencia, accesibilidad, escalabilidad, privacidad y sostenibilidad.

ANEXO II: REFLEXIONES ÉTICAS Y SOCIALES

Durante el desarrollo del presente Trabajo de Fin de Grado se ha considerado de forma específica el impacto ético y social de las decisiones tomadas en la implementación del sistema. Si bien dichas reflexiones no se han recogido explícitamente en los capítulos principales de la memoria, este anexo expone de manera razonada y justificada los principales puntos de análisis.

En primer lugar, desde el plano ético, uno de los aspectos más relevantes ha sido la protección de la privacidad y confidencialidad de los datos tratados. En el contexto académico, muchas de las encuestas procesadas contienen valoraciones personales de los alumnos, comentarios manuscritos y posibles referencias a terceros. Ante esta realidad, la decisión de trabajar exclusivamente con modelos de lenguaje de código abierto ejecutados en local, sin conexión a servidores externos ni envío de datos a plataformas comerciales, ha sido una elección clave. Esta arquitectura garantiza que ninguna información sensible o clasificada sea expuesta o transferida a sistemas que no están bajo control directo del usuario, lo que refuerza el cumplimiento del principio de confidencialidad, especialmente relevante en entornos institucionales o militares como el CUD-ENM.

Por otra parte, desde el punto de vista social, el sistema desarrollado persigue un objetivo claro de reducción de la carga de trabajo manual del profesorado, automatizando tareas rutinarias como la transcripción y el análisis de encuestas. Esta mejora no solo libera tiempo para actividades de mayor valor académico (como la docencia o la investigación), sino que también puede favorecer la participación del alumnado, al transmitir una mayor sensación de escucha y análisis efectivo de sus opiniones. Asimismo, al permitir analizar grandes volúmenes de encuestas de forma estructurada y objetiva, se contribuye a una toma de decisiones pedagógicas basada en datos, promoviendo una mejora continua de la calidad educativa.

Finalmente, este trabajo contribuye también a una democratización del uso de la inteligencia artificial, al demostrar que es posible desarrollar herramientas avanzadas sin depender de recursos comerciales o servicios cerrados. Esto abre la puerta a que instituciones con recursos limitados puedan incorporar tecnología puntera de forma ética, controlada y sostenible, sin comprometer la seguridad de los datos ni generar desigualdades tecnológicas.

En conjunto, el diseño y ejecución de este proyecto ha estado guiado por una reflexión ética orientada a maximizar el beneficio social del sistema, respetar los principios de seguridad y privacidad, y promover el uso responsable y accesible de tecnologías emergentes.

ANEXO III: CÓDIGO PRINCIPAL: MAIN.PY

```
import subprocess
import sys

##### BLOQUE 1 #####

print("\nBuenos días, bienvenido al programa de análisis automático de encuestas académicas mediante el uso de LLM, ",end="")
comenzar_conversion = str(input("¿desea comezar? [y/n]: "))
while (comenzar_conversion != "y" and comenzar_conversion != "n"):
comenzar_conversion = str(input("\n¿Desea comezar? [y/n]: "))
if comenzar_conversion == "n":
    print("\n¡Hasta la próxima!\n")
    sys.exit()
else: print("\nCargando modelos de lenguaje...\n")

subprocess.run(["python3", "LLM_Llama3.py"])
subprocess.run(["python3", "LLM_DeepSeek.py"])

print("\nModelos cargados. Procedo con la conversión de los pdf a imágenes y texto.\n")

subprocess.run(["python3", "Bloque_1.py"])

##### BLOQUE 2 #####

comenzar_LLM = str(input("\nFinalizada conversión de los documentos con éxito. ¿Comenzamos con el análisis del texto?[y/n]: "))
while (comenzar_LLM != "y" and comenzar_LLM != "n"): comenzar_LLM =
str(input("\n¿Comenzamos con el análisis del texto?[y/n]: "))
if comenzar_LLM == "n":
    print("\n¡Hasta la próxima!\n")
    sys.exit()
else: print("\nDe acuerdo, procedo con el análisis de los textos mediante LLM.\n")

subprocess.run(["python3", "Bloque_2.py"])

##### BLOQUE 3 #####

comenzar_procesamiento_datos = str(input("\nFinalizado el análisis de los textos mediante LLM. ¿Comenzamos con el procesamiento de datos y conversión en tablas formato CSV?[y/n]: "))
while (comenzar_procesamiento_datos != "y" and comenzar_procesamiento_datos != "n"): comenzar_procesamiento_datos = str(input("\n¿Comenzamos con el procesamiento de datos y conversión en tablas formato CSV?[y/n]: "))
if comenzar_procesamiento_datos == "n":
    print("\n¡Hasta la próxima!\n")
    sys.exit()
else: print("\nDe acuerdo, procedo con procesamiento de datos y conversión en tablas formato CSV.\n")

subprocess.run(["python3", "Bloque_3.py"])

##### BLOQUE 4 #####

comenzar_creacion_graficas = str(input("\nFinalizado el el procesamiento de datos y conversión en tablas formato CSV. ¿Comenzamos con la creación de gráficas para el informe?[y/n]: "))
while (comenzar_creacion_graficas != "y" and comenzar_LLM != "n"): comenzar_LLM =
str(input("\n¿Comenzamos con la creación de gráficas para el informe?[y/n]: "))
if comenzar_creacion_graficas == "n":
```

```
print("\n;Hasta la próxima!\n")
sys.exit()
else: print("\nDe acuerdo, procedo con la creación de gráficas para el
informe.\n")

subprocess.run(["python3", "Bloque_4.py"])

##### BLOQUE 5 #####

comenzar_analisis_respuestas = str(input("\nFinalizada la creación de gráficos
para el informe. ¿Comenzamos con el análisis de conclusiones mediante LLM?[y/n]:
"))
while (comenzar_creacion_graficas != "y" and comenzar_LLM != "n"): comenzar_LLM =
str(input("\n¿Comenzamos con el análisis de conclusiones mediante LLM?[y/n]: "))
if comenzar_creacion_graficas == "n":
    print("\n;Hasta la próxima!\n")
    sys.exit()
else: print("\nDe acuerdo, procedo con el análisis de conclusiones mediante
LLM.\n")

subprocess.run(["python3", "Bloque_5.py"])

##### BLOQUE 6 #####

comenzar_analisis_respuestas = str(input("\nFinalizado el análisis de conclusiones
mediante LLM. ¿Comenzamos con la generación de informes en PDF?[y/n]: "))
while (comenzar_creacion_graficas != "y" and comenzar_LLM != "n"): comenzar_LLM =
str(input("\n¿Comenzamos con la generación de informes en PDF?[y/n]: "))
if comenzar_creacion_graficas == "n":
    print("\n;Hasta la próxima!\n")
    sys.exit()
else: print("\nla generación de informes en PDF.\n")

subprocess.run(["python3", "Bloque_6.py"])

print("\n\nFinalizado el proceso de generación de informes en PDF. Informes
guardados en la carpeta Informes_PDF en el directorio principal.")

##### FIN DEL PROGRAMA #####

print("\n\n\n*** FIN DEL PROGRAMA ***\n\n")
```

ANEXO IV: CÓDIGOS DE BLOQUES DE LA ARQUITECTURA

A4.1 Bloque_1.py: conversión de información del PDF al formato TXT

```
import os
from os import mkdir
from Aux_OCR import Pdfconverter

opt_Escaneado = str(input("¿Todas las carillas tienen texto?[y/n]: "))
while(opt_Escaneado != "y" and opt_Escaneado != "n"): opt_Escaneado =
str(input("¿Todas las carillas tienen texto?[y/n]: "))
if(opt_Escaneado == "n"):
    num_pages = int(input("\n¿Cuántas carillas sí tienen texto?: "))
    while (opt_Escaneado != "y" and opt_Escaneado != "n"): opt_Escaneado =
int(input("\n¿Cuántas carillas sí tienen texto?: "))
else: num_pages = 0

x = y = z = 0

mkdir('Encuestas_PNG')
mkdir('Encuestas_TXT')
all_items = os.listdir('Scanned_PDF')
cursos = sorted([item for item in all_items if
os.path.isdir(os.path.join('Scanned_PDF', item))]) #sorted para que estén
ordenados
Ncursos = len(cursos)
print("Hay registrados "+str(Ncursos-1)+" cursos y el formato original:")
#print(cursos)
if (Ncursos == 0): pass
for x in range(Ncursos):
    mkdir(f'Encuestas_PNG/{cursos[x]}')
    mkdir(f'Encuestas_TXT/{cursos[x]}')
    all_items = os.listdir(f'Scanned_PDF/{cursos[x]}')
    grupos = sorted([item for item in all_items if
os.path.isdir(os.path.join(f'Scanned_PDF/{cursos[x]}', item))])
    Ngrupos = len(grupos)
    print(" * El curso "+str(cursos[x])+" tiene registrados "+str(Ngrupos)+"
grupos:")
    #print(grupos)
    if (Ngrupos == 0): pass
    for y in range(Ngrupos):
        mkdir(f'Encuestas_PNG/{cursos[x]}/{grupos[y]}')
        mkdir(f'Encuestas_TXT/{cursos[x]}/{grupos[y]}')
        encuestas = sorted([os.path.splitext(archivo)[0] for _, _, archivos in
os.walk(f'Scanned_PDF/{cursos[x]}/{grupos[y]}') for archivo in archivos])
        Nencuestas = len(encuestas)
        print(" - El grupo "+str(grupos[y])+" tiene registradas
"+str(Nencuestas)+" encuestas:")
        if (Nencuestas == 0): pass
        for z in range(Nencuestas):
            nombre = cursos[x] + "_" + grupos[y] + "_" + encuestas[z]

Pdfconverter(f'Scanned_PDF/{cursos[x]}/{grupos[y]}/{encuestas[z]}.pdf',f'Encuestas
_PNG/{cursos[x]}/{grupos[y]}',f'Encuestas_TXT/{cursos[x]}/{grupos[y]}', nombre,
num_pages, encuestas[z])
            print("\n Terminada la " + encuestas[z] + ".pdf")
            z=z+1
        y=y+1
    x=x+1
```

A4.2 Bloque_2.py: extracción de encuestas del resultado empleando LLM

```
import os
from os import mkdir
from Aux_prompt import prompt_extraccion
from Aux_prompt import prompt_encabezado
from LLM_Llama3 import analisis_LLM
from LLM_DeepSeek import analisis_LLM

#Evitar fallos después:
Noriginales = 1
rutas_originales = []
ruta_prompt = r"Recursos/prompt.txt"

mkdir('Resultados_TXT')
all_items = os.listdir('Encuestas_TXT')
cursos = sorted([item for item in all_items if
os.path.isdir(os.path.join('Encuestas_TXT', item))])
Ncursos = len(cursos)
print("Hay registrados "+str(Ncursos-1)+" cursos y el formato original:")
if (Ncursos == 0): pass
for x in range(Ncursos):
    if (x != 0): mkdir(f'Resultados_TXT/{cursos[x]}') #No generar la carpeta C0
    all_items = os.listdir(f'Encuestas_TXT/{cursos[x]}')
    grupos = sorted([item for item in all_items if
os.path.isdir(os.path.join(f'Encuestas_TXT/{cursos[x]}', item))])
    print(grupos)
    Ngrupos = len(grupos)
    print(" * El curso " + str(cursos[x]) + " tiene registrados " + str(Ngrupos) +
" grupos:")
    if (Ngrupos == 0): pass
    for y in range(Ngrupos):
        if (x != 0): #Como no existe la carpeta C0 daría error al generar el txt
            ruta_resultados = "Resultados_TXT/" + cursos[x] + "/" + cursos[x] +
"_" + grupos[y] + "_" + "resultados.txt"
            with open(ruta_resultados, "w", encoding="utf-8") as file:
                pass
            encuestas = sorted([os.path.splitext(archivo)[0] for _, _, archivos in
os.walk(f'Encuestas_TXT/{cursos[x]}/{grupos[y]}') for archivo in archivos])
            Nencuestas = len(encuestas)
            if (x==0): Noriginales = len(encuestas)
            Nreal_encuestas = int(len(encuestas)/Noriginales)
            print(" - El grupo " + str(grupos[y]) + " tiene registradas " +
str(Nreal_encuestas) + " encuestas desglosadas en " + str(Nencuestas) + " txt:")
            if (Nencuestas == 0): pass
            for z in range(Nencuestas):
                if (x==0):
                    all_items = os.listdir(f'Encuestas_TXT/{cursos[x]}/{grupos[y]}')
                    #originales = sorted([item for item in all_items if
os.path.isdir(os.path.join(f'Encuestas_TXT/{cursos[x]}/{grupos[y]}', item))])
                    originales = sorted([os.path.splitext(archivo)[0] for _, _,
archivos in os.walk(f'Encuestas_TXT/{cursos[x]}/{grupos[y]}') for archivo in
archivos])

                    Noriginales = len(originales)
                    print(f"Encontradas " + str(Noriginales) + " caras de referencia:
" + str(originales))
                    rutas_originales = ["Encuestas_TXT/" + cursos[x] + "/" + grupos[y]
+ "/" + archivo + ".txt" for archivo in originales]
                    y = Ngrupos-1
                    z = Nencuestas-1

                    ruta_encabezado = r"Recursos/encabezado.txt"
                    with open(ruta_encabezado, "w", encoding="utf-8") as file:
```

```

        pass

        for p in range(Noriginales): #Aquí estoy creando el txt del
encabezado para usar luego en las tablas CSV

            prompt_encabezado(rutas_originales[p]) #genera prompt para
sacar encabezado del original
            respuesta_encabezado = analisis_LLM(ruta_prompt) #recibe
respuesta de encabezado del original

            with open(ruta_encabezado, "a", encoding="utf-8") as file:
                file.write(respuesta_encabezado) #escribe en el txt de
encabezado

            print("Creado encabezado sin novedad, guardado en " +
ruta_encabezado)

        else:
            #número de cara de la encuesta:
            num_cara = z % Noriginales

            #ruta de la hoja de la encuesta que toca analizar:
            ruta_txt = "Encuestas_TXT/" + cursos[x] + "/" + grupos[y] + "/" +
encuestas[z] + ".txt" #ruta al txt que estamos analizando
            ruta_original = rutas_originales[num_cara] #ruta al txt de
referencia de la página que corresponde

            prompt_extraccion(ruta_original, ruta_txt)
            #Creado prompt.txt con el original y el de la encuesta que toca y
guardado en Recursos/prompt.txt

            respuesta = analisis_LLM(ruta_prompt)

            pre_respuesta = cursos[x] + ";" + grupos[y] + ";encuesta" +
str(int(z/Noriginales)+1) + ";"

            #Ha activado el análisis del LLM que devuelve en respuesta las
respuestas de la encuesta en el formato
            with open(ruta_resultados, "a", encoding="utf-8") as file:
                if (num_cara == 0): file.write("\n"+pre_respuesta+respuesta)
                else: file.write(respuesta)

            z=z+1
            print("\n      Terminado el grupo " + grupos[y] + ".\n")
            y=y+1
            x=x+1

```

A4.3 Bloque_3.py: conversión de resultados a CSV y tratamiento de datos

```

import os
from os import mkdir
from Aux_CSV import tabla_csv_grupo
from Aux_CSV import txt_sum_grupo
from Aux_CSV import recopilatorio_datos

ruta_encabezado = r"Recursos/encabezado.txt"

##### PROCESAMIENTO POR GRUPOS #####
mkdir('Grupos_CSV') #Almacena resultados csv por grupo dentro de cada curso
mkdir('Cursos_preCSV_TXT') #Almacena resultados txt por curso
all_items = os.listdir('Resultados_TXT')

```

```

cursos = sorted([item for item in all_items if
os.path.isdir(os.path.join('Resultados_TXT', item))])
Ncursos = len(cursos)
print("Hay registrados "+str(Ncursos)+" cursos: ")
if (Ncursos == 0): pass
for x in range(Ncursos):
    mkdir(f'Grupos_CSV/{cursos[x]}')
    ruta_resultados_resumen_txt = "Cursos_preCSV_TXT/" + cursos[x] +
"_resultados.txt"
    with open(ruta_resultados_resumen_txt, "w", encoding="utf-8") as file:
        pass
    #Los CSV se crean directamente en la función
    txt_resultados = sorted([os.path.splitext(archivo)[0] for _, _, archivos in
os.walk(f'Resultados_TXT/{cursos[x]}') for archivo in archivos])
    Ntxt_resultados = len(txt_resultados)
    print(" * El curso " + str(cursos[x]) + " tiene registrados " +
str(Ntxt_resultados) + " resultados grupales.")
    if (Ntxt_resultados == 0): pass
    for y in range(Ntxt_resultados):
        ruta_txt_grupo = "Resultados_TXT/" + cursos[x] + "/" + txt_resultados[y] +
".txt" #recuperamos la ruta del txt a procesar
        ruta_resultados_grupo = "Grupos_CSV/" + cursos[x] + "/" +
txt_resultados[y] + ".csv" #creamos la ruta del csv donde irá la tabla de
resultados procesados
        # Los CSV se crean directamente en la función
        tabla_csv_grupo(ruta_encabezado, ruta_txt_grupo, ruta_resultados_grupo)
#Se llama a función para procesar. Se le entrega la ruta del txt donde está el
encabezado de las columnas de la tabla, la ruta del txt a procesar y la del csv de
resultados procesados
        print("\n Terminado el grupo " + txt_resultados[y] + ".\n")

        resultados = txt_sum_grupo(ruta_resultados_grupo)
        pre_resultados = txt_resultados[y] + ";"
        with open(ruta_resultados_resumen_txt, "a", encoding="utf-8") as file:
            file.write("\n" + pre_resultados + resultados)
        print("Guardado en " + ruta_resultados_resumen_txt)

        y=y+1
    print("\n Terminado el curso " + str(cursos[x]) + ".")
    x=x+1

##### PROCESAMIENTO POR CURSOS #####
mkdir('Cursos_CSV') #Almacena resultados csv por curso
mkdir('Total_preCSV_TXT') #Almacena resultados txt totales
resumenes_txt = sorted([os.path.splitext(archivo)[0] for _, _, archivos in
os.walk('Cursos_preCSV_TXT') for archivo in archivos])
Nresumenes_txt = len(resumenes_txt)
print("Hay registrados "+str(resumenes_txt)+" documentos")
ruta_resultados_totales_txt = "Total_preCSV_TXT/Resultados_CSV_totales.txt"
with open(ruta_resultados_totales_txt, "w", encoding="utf-8") as file:
    pass

if (Nresumenes_txt == 0): pass
for z in range(Nresumenes_txt):
    ruta_txt_curso = "Cursos_preCSV_TXT/" + resumenes_txt[z] + ".txt" #
recuperamos la ruta del txt a procesar
    ruta_resultados_grupo = "Cursos_CSV/" + resumenes_txt[z] + ".csv" # creamos
la ruta del csv donde irá la tabla de resultados procesados
    # Los CSV se crean directamente en la función
    recopilatorio_datos(ruta_encabezado, ruta_txt_curso, ruta_resultados_grupo) #
Se llama a función para procesar. Se le entrega la ruta del txt donde está el
encabezado de las columnas de la tabla, la ruta del txt a procesar y la del csv de
resultados procesados

```

```

print("\n      Terminado el grupo " + resmenes_txt[z] + ".\n")

resultados2 = txt_sum_grupo(ruta_resultados_grupo)
pre_resultados2 = resmenes_txt[z] + ";"
with open(ruta_resultados_totales_txt, "a", encoding="utf-8") as file:
    file.write("\n" + pre_resultados2 + resultados2)
print("Guardado en " + ruta_resultados_totales_txt)

z = z + 1

##### PROCESAMIENTO DATOS TOTAL #####
mkdir('Total_CSV')
ruta_csv_total = "Total_CSV/Total.csv"
recopilatorio_datos(ruta_encabezado, ruta_resultados_totales_txt, ruta_csv_total)

print("\n\nGuardado recopilatorio total en " + ruta_csv_total)

```

A4.4 Bloque_4.py: creación de gráficas de resultados

```

import os
from os import mkdir
from Aux_prompt import prompt_extraccion
from Aux_prompt import prompt_encabezado
from LLM_Llama3 import analisis_LLM
from LLM_DeepSeek import analisis_LLM

#Evitar fallos después:
Noriginales = 1
rutas_originales = []
ruta_prompt = r"Recursos/prompt.txt"

mkdir('Resultados_TXT')
all_items = os.listdir('Encuestas_TXT')
cursos = sorted([item for item in all_items if
os.path.isdir(os.path.join('Encuestas_TXT', item))])
Ncursos = len(cursos)
print("Hay registrados "+str(Ncursos-1)+" cursos y el formato original:")
if (Ncursos == 0): pass
for x in range(Ncursos):
    if (x != 0): mkdir(f'Resultados_TXT/{cursos[x]}') #No generar la carpeta C0
    all_items = os.listdir(f'Encuestas_TXT/{cursos[x]}')
    grupos = sorted([item for item in all_items if
os.path.isdir(os.path.join(f'Encuestas_TXT/{cursos[x]}', item))])
    print(grupos)
    Ngrupos = len(grupos)
    print(" * El curso " + str(cursos[x]) + " tiene registrados " + str(Ngrupos) +
" grupos:")
    if (Ngrupos == 0): pass
    for y in range(Ngrupos):
        if (x != 0): #Como no existe la carpeta C0 daría error al generar el txt
            ruta_resultados = "Resultados_TXT/" + cursos[x] + "/" + cursos[x] +
"_" + grupos[y] + "_" + "resultados.txt"
            with open(ruta_resultados, "w", encoding="utf-8") as file:
                pass
            encuestas = sorted([os.path.splitext(archivo)[0] for _, _, archivos in
os.walk(f'Encuestas_TXT/{cursos[x]}/{grupos[y]}') for archivo in archivos])
            Nencuestas = len(encuestas)
            if (x==0): Noriginales = len(encuestas)
            Nreal_encuestas = int(len(encuestas)/Noriginales)
            print(" - El grupo " + str(grupos[y]) + " tiene registradas " +
str(Nreal_encuestas) + " encuestas desglosadas en " + str(Nencuestas) + " txt:")
            if (Nencuestas == 0): pass
            for z in range(Nencuestas):

```

```

        if (x==0):
            all_items = os.listdir(f'Encuestas_TXT/{cursos[x]}/{grupos[y]}')
            #originales = sorted([item for item in all_items if
os.path.isdir(os.path.join(f'Encuestas_TXT/{cursos[x]}/{grupos[y]}', item))])
            originales = sorted([os.path.splitext(archivo)[0] for _, _,
archivos in os.walk(f'Encuestas_TXT/{cursos[x]}/{grupos[y]}') for archivo in
archivos])

            Nororiginales = len(originales)
            print(f"Encontradas " + str(Nororiginales) + " caras de referencia:
" + str(originales))
            rutas_originales = ["Encuestas_TXT/" + cursos[x] + "/" + grupos[y]
+ "/" + archivo + ".txt" for archivo in originales]
            y = Ngrupos-1
            z = Nencuestas-1

            ruta_encabezado = r"Recursos/encabezado.txt"
            with open(ruta_encabezado, "w", encoding="utf-8") as file:
                pass

            for p in range(Noriginales): #Aquí estoy creando el txt del
encabezado para usar luego en las tablas CSV

                prompt_encabezado(rutas_originales[p]) #genera prompt para
sacar encabezado del original
                respuesta_encabezado = analisis_LLM(ruta_prompt) #recibe
respuesta de encabezado del original

                with open(ruta_encabezado, "a", encoding="utf-8") as file:
                    file.write(respuesta_encabezado) #escribe en el txt de
encabezado

            print("Creado encabezado sin novedad, guardado en " +
ruta_encabezado)

        else:
            #número de cara de la encuesta:
            num_cara = z % Nororiginales

            #ruta de la hoja de la encuesta que toca analizar:
            ruta_txt = "Encuestas_TXT/" + cursos[x] + "/" + grupos[y] + "/" +
encuestas[z] + ".txt" #ruta al txt que estamos analizando
            ruta_original = rutas_originales[num_cara] #ruta al txt de
referencia de la página que corresponde

            prompt_extraccion(ruta_original, ruta_txt)
            #Creado prompt.txt con el original y el de la encuesta que toca y
guardado en Recursos/prompt.txt

            respuesta = analisis_LLM(ruta_prompt)

            pre_respuesta = cursos[x] + ";" + grupos[y] + ";encuesta" +
str(int(z/Noriginales)+1) + ";"

            #Ha activado el análisis del LLM que devuelve en respuesta las
respuestas de la encuesta en el formato
            with open(ruta_resultados, "a", encoding="utf-8") as file:
                if (num_cara == 0): file.write("\n"+pre_respuesta+respuesta)
                else: file.write(respuesta)

            z=z+1
            print("\n      Terminado el grupo " + grupos[y] + ".\n")
            y=y+1
            x=x+1

```

A4.5 Bloque_5.py: análisis de resultados mediante LLM

```

import os
from os import mkdir
from Aux_CSV import transformar_csv_a_txt
from Aux_prompt import prompt_analisis
from Aux_prompt import prompt_conclusion
from LLM_Llama3 import analisis_LLM
from LLM_DeepSeek import analisis_LLM

ruta_encuesta_original = r"Recursos/encuesta_original.txt"
ruta_prompt = r"Recursos/prompt.txt"

##### ANÁLISIS POR GRUPOS #####
print("\n\nEmpezamos con el estudio por grupos:\n")
mkdir('Respuestas_Grupos_txt') #Almacena los txt que resumen las respuestas de
cada grupo
mkdir('Conclusiones_Grupos_LLM') #Almacena los txt que resumen las respuestas de
cada grupo
all_items = os.listdir('Grupos_CSV')
cursos = sorted([item for item in all_items if
os.path.isdir(os.path.join('Grupos_CSV', item))])
Ncursos = len(cursos)
print("Hay registrados "+str(Ncursos)+" cursos: ")
if (Ncursos == 0): pass
for x in range(Ncursos):
    mkdir(f'Respuestas_Grupos_txt/{cursos[x]}')
    mkdir(f'Conclusiones_Grupos_LLM/{cursos[x]}')
    csv_resultados = sorted([os.path.splitext(archivo)[0] for _, _, archivos in
os.walk(f'Grupos_CSV/{cursos[x]}') for archivo in archivos])
    Ntxt_resultados = len(csv_resultados)
    print(" * El curso " + str(cursos[x]) + " tiene registrados " +
str(Ntxt_resultados) + " resultados grupales.")
    if (Ntxt_resultados == 0): pass
    for y in range(Ntxt_resultados):

        ruta_resultados_grupo_csv = "Grupos_CSV/" + cursos[x] + "/" +
csv_resultados[y] + ".csv" # recuperamos la ruta del csv donde está la tabla de
resultados procesados
        ruta_resultados_grupo_txt = "Respuestas_Grupos_txt/" + cursos[x] + "/" +
csv_resultados[y] + ".txt" #creamos rutas para guardar las respuestas por líneas
        with open(ruta_resultados_grupo_txt, "w", encoding="utf-8") as file:
            pass

        ruta_conclusion_txt = "Conclusiones_Grupos_LLM/" + cursos[x] + "/" +
csv_resultados[y] + ".txt" #ruta para el txt que almacene las conclusiones por
líneas
        with open(ruta_conclusion_txt, "w", encoding="utf-8") as file:
            pass

        transformar_csv_a_txt(ruta_resultados_grupo_csv,
ruta_resultados_grupo_txt) #se llama a la función para transofrmar el csv del
grupo en el txt con respuestas por línea

        with open(ruta_resultados_grupo_txt, "r", encoding="utf-8") as file:
            num_lineas_txt = sum(1 for line in file) #sacamos el numero de lineas
de ese txt, que deberían de ser 13 porque es el numero de preguntas

        for z in range(num_lineas_txt):

            prompt_analisis(ruta_encuesta_original, ruta_resultados_grupo_txt, z)
            #generado prompt con la pregunta y respuestas de la primera fila del
orginial y con la primera fila de las respuestas

```

```

conclusion = analisis_LLM(ruta_prompt)

#se genera una conclusión automática para esos resultados

with open(ruta_conclusion_txt, "a", encoding="utf-8") as file:
    guardar = "Pregunta " + str(z+1) + ";" + conclusion + "\n"
    file.write(guardar) #escribe en el txt de encabezado #añadimos
la conclusión a la línea correspondiente

#Ahora guardamos en la última línea del txt la conclusión final de las
otras generadas:
prompt_conclusion(ruta_conclusion_txt) #prompt generado con las respuestas
anteriores

conclusion_final = analisis_LLM(ruta_conclusion_txt)
#conclusion_final = "Conclusión final."

with open(ruta_conclusion_txt, "a", encoding="utf-8") as file:
    guardar = "Conclusión final;" + conclusion_final
    file.write(guardar) # escribe en el txt de encabezado #añadimos la
conclusión a la línea correspondiente

y=y+1
print("\n Terminado el curso " + str(cursos[x]) + ".")
x=x+1

##### ANÁLISIS POR CURSOS #####
print("\n\nAhora vamos con el estudio por cursos:\n")
mkdir('Respuestas_Cursos_txt') #Almacena los txt que resumen las respuestas de
cada curso
mkdir('Conclusiones_Cursos_LLM') #Almacena los txt que resumen las respuestas de
cada curso

resumenes_csv = sorted([os.path.splitext(archivo)[0] for _, _, archivos in
os.walk('Cursos_CSV') for archivo in archivos])
Nresumenes_csv = len(resumenes_csv)
print("Hay registrados "+str(Nresumenes_csv)+" documentos")
if (Nresumenes_csv == 0): pass
for a in range(Nresumenes_csv):

    ruta_resultados_curso_csv = "Cursos_CSV/" + resumenes_csv[a] + ".csv" #
recuperamos la ruta del csv donde está la tabla de resultados procesados
    ruta_resultados_curso_txt = "Respuestas_Cursos_txt/" + resumenes_csv[a] +
".txt" # creamos rutas para guardar las respuestas por líneas
    with open(ruta_resultados_curso_txt, "w", encoding="utf-8") as file:
        pass

    ruta_conclusion_txt = "Conclusiones_Cursos_LLM/" + resumenes_csv[a] + ".txt"
# ruta para el txt que almacene las conclusiones por líneas
    with open(ruta_conclusion_txt, "w", encoding="utf-8") as file:
        pass

    transformar_csv_a_txt(ruta_resultados_curso_csv,ruta_resultados_curso_txt) #
se llama a la función para transofrmar el csv del grupo en el txt con respuestas
por línea

    with open(ruta_resultados_curso_txt, "r", encoding="utf-8") as file:
        num_lineas_txt = sum(1 for line in file) # sacamos el numero de lineas de
ese txt, qué deberían de ser 13 porque es el numero de preguntas

    for b in range(num_lineas_txt):

        prompt_analisis(ruta_encuesta_original, ruta_resultados_curso_txt, b)

```

```
# generado prompt con la pregunta y respuestas de la primera fila del
orginial y con la primera fila de las respuestas

conclusion = analisis_LLM(ruta_prompt)

# se genera una conclusión automática para esos resultados

with open(ruta_conclusion_txt, "a", encoding="utf-8") as file:
    guardar = "Pregunta " + str(b+1) + ";" + conclusion + "\n"
    file.write(guardar) # escribe en el txt de encabezado #añadimos la
conclusión a la línea correspondiente

b=b+1

# Ahora guardamos en la última línea del txt la conclusión final de las otras
generadas:
prompt_conclusion(ruta_conclusion_txt) # prompt generado con las respuestas
anteriores

conclusion_final = analisis_LLM(ruta_conclusion_txt)
#conclusion_final = "Conclusión final."

with open(ruta_conclusion_txt, "a", encoding="utf-8") as file:
    guardar = "Conlusión final;" + conclusion_final
    file.write(guardar) # escribe en el txt de encabezado #añadimos la
conclusión a la línea correspondiente

a=a+1

##### ANÁLISIS DATOS TOTAL #####
print("\n\nTerminamos con el estudio total:\n")
mkdir('Respuestas_Total_txt') #Almacena los txt que resumen las respuestas totales
mkdir('Conclusiones_Total_LLM') #Almacena los txt que resumen las respuestas
totales

ruta_resultados_total_csv = "Total_CSV/Total.csv"
ruta_resultados_total_txt = "Respuestas_Total_txt/Respuestas_Total.txt"
with open(ruta_resultados_total_txt, "w", encoding="utf-8") as file:
    pass

ruta_conclusion_txt = "Conclusiones_Total_LLM/Conclusiones_Total.txt" # ruta para
el txt que almacene las conclusiones por líneas
with open(ruta_conclusion_txt, "w", encoding="utf-8") as file:
    pass

transformar_csv_a_txt(ruta_resultados_total_csv, ruta_resultados_total_txt) # se
llama a la función para transofrmar el csv del grupo en el txt con respuestas por
línea

with open(ruta_resultados_total_txt, "r", encoding="utf-8") as file:
    num_lineas_txt = sum(1 for line in file) # sacamos el numero de lineas de ese
txt, qué deberían de ser 13 porque es el numero de preguntas

for c in range(num_lineas_txt):

    prompt_analisis(ruta_encuesta_original, ruta_resultados_total_txt, c)
    # generado prompt con la pregunta y respuestas de la primera fila del orginial
y con la primera fila de las respuestas

    #conclusion = "Conclusión pregunta " + str(c+1) + "."
    conclusion = analisis_LLM(ruta_prompt)

# se genera una conclusión automática para esos resultados
```

```

with open(ruta_conclusion_txt, "a", encoding="utf-8") as file:
    guardar = "Pregunta " + str(c+1) + ";" + conclusion + "\n"
    file.write(guardar) # escribe en el txt de encabezado #añadimos la
conclusión a la línea correspondiente

c=c+1

# Ahora guardamos en la última línea del txt la conclusión final de las otras
generadas:
prompt_conclusion(ruta_conclusion_txt) # prompt generado con las respuestas
anteriores

conclusion_final = analisis_LLM(ruta_conclusion_txt)
#conclusion_final = "Conclusión final."

with open(ruta_conclusion_txt, "a", encoding="utf-8") as file:
    guardar = "Conclusión final;" + conclusion_final
    file.write(guardar) # escribe en el txt de encabezado #añadimos la
conclusión a la línea correspondiente

print("\n\nTerminadas las conclusiones.\n")

```

A4.6 Bloque_6.py: generación automática de informes en PDF

```

import os
from os import mkdir
from Aux_informe import generar_pdf

ruta_encuesta_original = r"Recursos/encuesta_original.txt"
ruta_prompt = r"Recursos/prompt.txt"

eleccion = input("\n\nEmpezamos con la generación de informes, indíquenos qué
informes quiere generar:\n\n a) Informes por grupos.\n b) Informes por
cursos.\n c) El informe total.\n d) Los informes por cursos y el informe
total.\n\n¿Qué opción elige?[a/b/c/d]: ").lower()
while (eleccion != "a" and eleccion != "b" and eleccion != "c" and eleccion !=
"d"): eleccion = str(input("\n¿Qué opción elige?[a/b/c/d]: ")).lower()

format = input("Existen estos formatos para las gráficas:\n\n a) Barras\n b)
Circular (pastel)\n\n¿Cuál prefieres?[a/b]: ").lower()
while (format != "a" and format != "b"): eleccion = str(input("\n¿Cuál
prefieres?[a/b]: ")).lower()

mkdir('Informes_PDF') #Almacena los informes generados

##### CREACIÓN DE INFORMES POR GRUPO #####
if (eleccion == "a"):
    all_items = os.listdir('Conclusiones_Grupos_LLM')
    cursos = sorted([item for item in all_items if
os.path.isdir(os.path.join('Conclusiones_Grupos_LLM', item))])
    Ncursos = len(cursos)
    print("Hay registrados "+str(Ncursos)+" cursos")

    all_items = os.listdir('Graficas_Grupos')
    cursos_graf = sorted([item for item in all_items if
os.path.isdir(os.path.join('Graficas_Grupos', item))])
    Ncursos_graf = len(cursos_graf)
    print("\nHay registrados " + str(Ncursos_graf) + " cursos con gráficas: ")

    mkdir('Informes_PDF/Grupos') #Almacena los informes generados por grupos si se
ha elegido esa opción

```

```

    if (Ncursos == 0 or eleccion != "a"): pass
    for x in range(Ncursos):
        mkdir(f'Informes_PDF/Grupos/{cursos[x]}')
        txt_conclusiones = sorted([os.path.splitext(archivo)[0] for _, _, archivos
in os.walk(f'Conclusiones_Grupos_LLM/{cursos[x]}') for archivo in archivos])
        Ntxt_conclusiones = len(txt_conclusiones)
        print(" * El curso " + str(cursos[x]) + " tiene registrados " +
str(Ntxt_conclusiones) + " resultados grupales.")

        all_items = os.listdir(f'Graficas_Grupos/{cursos[x]}')
        grupos_graf = sorted([item for item in all_items if
os.path.isdir(os.path.join(f'Graficas_Grupos/{cursos[x]}', item))])
        Ngrupos_graf = len(grupos_graf)
        print("\nHay registrados " + str(Ngrupos_graf) + " grupos con gráficas")

        if (Ntxt_conclusiones == 0): pass
        for y in range(Ntxt_conclusiones):

            asignatura = ["Informática para la Ingeniería", "Fundamentos de
Electrotecnia", "Tecnología Electrónica", "Fundamentos de Automática",
"Fundamentos de Redes de Ordenadores"]

            titulo_informe = "Informe de Encuestas 2025 - " + asignatura[y] + " -
Grupo " + str(y+1)
            #ruta_encuesta_original
            ruta_grafica = "Graficas_Grupos/" + cursos_graf[x] + "/" +
grupos_graf[y]
            ruta_txt_conclusiones = "Conclusiones_Grupos_LLM/" + cursos[x] + "/" +
txt_conclusiones[y] + ".txt"
            ruta_pdf_informe = "Informes_PDF/Grupos/" + cursos[x] +
"/Informe_encuesta_" + cursos[x] + "_G" + str(y+1) + ".pdf"
            #format de gráfico

            generar_pdf(titulo_informe, ruta_encuesta_original, ruta_grafica,
ruta_txt_conclusiones, ruta_pdf_informe, format)

            print("Generado informe en PDF del grupo " + str(y+1) + "del curso " +
cursos[x])

            y=y+1
        print("\n Terminado el curso " + str(cursos[x]) + ".")
        x=x+1

##### CREACIÓN DE INFORMES POR CURSOS #####
if (eleccion == "b" or eleccion == "d"):

    txt_conclusiones = sorted([os.path.splitext(archivo)[0] for _, _, archivos in
os.walk('Conclusiones_Cursos_LLM') for archivo in archivos])
    Ntxt_conclusiones = len(txt_conclusiones)
    print("Hay registradas "+str(txt_conclusiones)+ " conclusiones de curso")

    all_items = os.listdir(f'Graficas_Cursos')
    cursos_graf = sorted([item for item in all_items if
os.path.isdir(os.path.join(f'Graficas_Cursos', item))])
    Ncursos_graf = len(cursos_graf)
    print("\nHay registrados " + str(Ncursos_graf) + " cursos con gráficas")

    mkdir('Informes_PDF/Cursos') #Almacena los informes generados por grupos si se
ha elegido esa opción

    if (Ntxt_conclusiones == 0 or eleccion != "b" or eleccion != "d"): pass
    for a in range(Ntxt_conclusiones):

```

```

    asignatura = ["Informática para la Ingeniería", "Fundamentos de
Electrotecnia", "Tecnología Electrónica", "Fundamentos de Automática",
"Fundamentos de Redes de Ordenadores"]

    titulo_informe = "Informe de Encuestas 2025 - " + asignatura[a]
    # ruta_encuesta_original
    ruta_grafica = "Graficas_Cursos/" + cursos_graf[a]
    ruta_txt_conclusiones = "Conclusiones_Cursos_LLM/" + txt_conclusiones[a] +
".txt"
    ruta_pdf_informe = "Informes_PDF/Cursos/Informe_encuesta_" +
cursos_graf[a] + ".pdf"
    #format de gráfico

    generar_pdf(titulo_informe, ruta_encuesta_original, ruta_grafica,
ruta_txt_conclusiones, ruta_pdf_informe, format)

    print("Generado informe en PDF del curso " + cursos_graf[a])

    a=a+1

##### CREACIÓN DE INFORME DATOS TOTALES #####
if (eleccion == "c" or eleccion == "d"):

    txt_conclusiones = sorted([os.path.splitext(archivo)[0] for _, _, archivos in
os.walk('Conclusiones_Total_LLM') for archivo in archivos])
    Ntxt_conclusiones = len(txt_conclusiones)
    print("Hay registrada " + str(txt_conclusiones) + " conclusión total")

    mkdir('Informes_PDF/Total') #Almacena los informes generados por grupos si se
ha elegido esa opción

    titulo_informe = "Informe de Encuestas 2025 - Conclusiones globales"
    # ruta_encuesta_original
    ruta_grafica = "Graficas_Total"
    ruta_txt_conclusiones = "Conclusiones_Total_LLM/" + txt_conclusiones[0] +
".txt"
    ruta_pdf_informe = "Informes_PDF/Total/Informe_encuesta_Total.pdf"
    #format de gráfico

    generar_pdf(titulo_informe, ruta_encuesta_original, ruta_grafica,
ruta_txt_conclusiones, ruta_pdf_informe, format)

    print("Generado informe global en PDF")

print("\n\nCerrado programa de generación de informes, terminados todos los informes
solicitados.\n")

```

ANEXO V: CÓDIGOS DE MÓDULOS AUXILIARES

A5.1 Aux_OCR.py: código auxiliar para convertir PDF en TXT

```
from pdf2image import convert_from_path
import pytesseract

#función encargada de convertir un PDF enrutado en imágenes PNG y en textos TXT
(tras su analizado en Tesseract-OCR)

def Pdfconverter(ruta_inicial, ruta_final, ruta_txt_final, Nombre_doc, Npages,
Nombre_pdf):
    pages = convert_from_path(ruta_inicial, 500)
    if (Npages == 0): Npages = len(pages)
    i=0
    for page in pages:
        i=i+1
        nueva_ruta = ruta_final + "/" + str(Nombre_doc) + "_" + str(i) + ".png"
        page.save(nueva_ruta, 'PNG')
        print("    -->Convertido el documento: " + str(Nombre_pdf) + ".pdf en la
imagen: " + str(Nombre_doc) + "_" + str(i) + ".png",end="")
        if (i <= Npages):
            custom_config = r'--oem 3 --psm 6'
            extracted_text = pytesseract.image_to_string(nueva_ruta,
config=custom_config, lang="eng+spa")
            print(" y en el texto: " + str(Nombre_doc) + "_" + str(i) + ".txt")
            txt_name = ruta_txt_final + "/" + str(Nombre_doc) + "_" + str(i) +
".txt"
            with open(txt_name, "w", encoding="utf-8") as file:
file.write(extracted_text)
```

A5.2 Aux_prompt.py: código auxiliar para generar *prompts* automáticamente

```
def prompt_extraccion(ruta_reference, ruta_error):

    prompt1 = """I will provide you with two texts. The first one, which I will
call REFERENCE_TEXT, will contain the text you must use as a reference. The second
one will be called ERROR_TEXT and contains a very similar text but with some
differences. It is a text composed of questions numbered with numbers and answers
labeled with letters. Your goal is to detect, for each question number, which
answer subsection contains the error. The only errors you must consider are those
within the answers. There can be more than one subsection with an error per
question. Only changes in the first word of each answer subsection should be
considered as errors. You must not mark as an error any changes within the content
of the subsection, even if there are spelling errors, typographical mistakes, or
modifications anywhere else within the text that is not its first word. The first
word of the subsection is understood to be the identifier (for example: 'a)',
'b)', etc.). As long as the identifier is correct, there will be no error. Do not
disregard changes to special characters such as '@' or '('. The only character you
must disregard if detected is the character '|'. That is, if you detect the
character '|', ignore it as if it weren't there. You must not mark as an error any
changes occurring after the words "Otro/s" and "Otra/s", regardless of their
nature. This means any modification, addition, or deletion of characters after
these words must be completely ignored in your analysis, even if they involve
special characters like '(', ')', '_', '-', or spaces. Remember that I'm only
interested in errors in the first word of each answer subsection, as those are the
ones you must indicate to me. If the answer identifier (example: 'b)') is correct
and the change occurs in the word immediately following it, it will not be
considered an error. However, if the identifier has been modified in any way (for
example: 'c)') instead of 'c)'), that will indeed be considered an error. Remember
not to consider as errors any spelling errors, typographical mistakes, or
modifications in any part of the text other than its first word. For instance, if
```

in a word that isn't the first word there is an error due to an accent mark, it will not be considered an error. For example: if instead of 'código' it says 'codigo', it will not be considered an error. Remember that if the first word is the subsection identifier and it has no errors, you must not count it as an error. For example: if the subsection begins with 'a) Sf. Antes', even if 'Sf.' wasn't in the original text, since it doesn't alter the first word ('a)'), there's no error in that subsection.

You will indicate the errors to me in the following format: 'x:y;x:y...', without line breaks or separation across multiple lines, such that 'x' corresponds to the question number, and 'y' to the letter (the subsection) of the response where you've detected the error. If there is more than one subsection with an error in a question, list all the erroneous subsections separated by commas. For example: '1:d;2:a;3:b,c'. In this example, an error was detected in subsection 'd' of question 1, an error in subsection 'a' of question 2, and errors in subsections 'b' and 'c' of question 3. If no errors are detected in a question, use the character 'z' as the value of 'y'. Therefore, your response must consist solely of a single line of numbers and letters as I've specified. I now proceed to provide the texts:

```
REFERENCE_TEXT: """
```

```
with open(ruta_reference, "r", encoding="utf-8") as file:
    REFERENCE_TEXT = file.read()
```

```
prompt2 = """
```

```
ERROR_TEXT: """
```

```
with open(ruta_error, "r", encoding="utf-8") as file:
    ERROR_TEXT = file.read()
```

```
prompt3 = """
```

Analyze the texts, look for errors, and give me the answer exactly as I've requested. Give me only the answer I've requested. Give me the answer in one line. Don't show me the prompt I've written.'"""

```
prompt = prompt1 + REFERENCE_TEXT + prompt2 + ERROR_TEXT + prompt3
```

```
ruta_archivo = r"Recursos/prompt.txt"
```

```
with open(ruta_archivo, "w", encoding="utf-8") as file:
    file.write(prompt)
```

```
def prompt_encabezado(ruta_reference):
```

```
    pregunta = "In this text, there is a questionnaire consisting of questions and
    corresponding answers. I need you to clearly distinguish between questions and
    their respective answer choices. List all answer choices using the following
    format: '1a;1b;2a;2b;2c;3a;...;' where each number represents the question number,
    and each letter represents a specific answer choice within that question.
    Additionally, for every question, include an extra answer choice labeled with the
    letter 'z'. You must add this 'z' option even to those questions that originally
    do not have multiple-choice answer options. Please present everything in a single
    line. The text is: \n"
```

```
with open(ruta_reference, "r", encoding="utf-8") as file:
    texto = file.read()
```

```
prompt = pregunta + texto
```

```
ruta_archivo = r"Recursos/prompt.txt"

with open(ruta_archivo, "w", encoding="utf-8") as file:
    file.write(prompt)

def prompt_analisis(ruta_preguntas, ruta_respuestas, num_linea):

    prompt1 = "For the next question: "
    with open(ruta_preguntas, "r", encoding="utf-8") as file:
        for i, linea in enumerate(file, start=0):
            if i == num_linea:
                PREGUNTA = linea.strip() # Devuelve la línea sin saltos de línea
    prompt2 = "'\n\nThe answers received are: "
    with open(ruta_respuestas, "r", encoding="utf-8") as file:
        for i, linea in enumerate(file, start=0):
            if i == num_linea:
                RESPUESTAS = linea.strip() # Devuelve la línea sin saltos de
línea
    prompt3 = "'\n\nPrepare an analysis in Spanish of those answers and give it to
me. Just give me the analysis in one paragraph. Don't show me the prompt I've
written."

    prompt = prompt1 + PREGUNTA + prompt2 + RESPUESTAS + prompt3

    ruta_archivo = r"Recursos/prompt.txt"

    with open(ruta_archivo, "w", encoding="utf-8") as file:
        file.write(prompt)

def prompt_conclusion(ruta_analisis):

    prompt1 = "I am going to provide you with the analyses in Spanish of the
different responses from a survey. I want you to read everything and give me a
final analysis in the form of a conclusion that summarizes the general ideas
expressed in the survey responses. I want the final conclusion on the survey's
topic of study to be clearly stated and explained. The analyses, which are
numbered according to each question, are as follows: \n\n"

    with open(ruta_analisis, "r", encoding="utf-8") as file:
        PREGUNTA = file.read()

    prompt2 = "'\n\nI want you to generate only one paragraph in Spanish with the
complete conclusion analysis."

    prompt = prompt1 + PREGUNTA + prompt2

    ruta_archivo = r"Recursos/prompt.txt"

    with open(ruta_archivo, "w", encoding="utf-8") as file:
        file.write(prompt)
```

A5.3 Aux_CSV.py: código auxiliar para generar tablas CSV

```
import pandas as pd

def tabla_csv_grupo (archivo_encabezado, archivo_datos, csv_resultados): #para
sacar los datos de los txt de Encuestas_txt
    # Leer el encabezado del archivo
    with open(archivo_encabezado, 'r', encoding='utf-8') as file:
        encabezado = file.readline().strip().split(';') # Leer y separar
encabezado
```

```

# Crear una lista vacía para almacenar las filas de datos
datos_procesados = []

# Leer los datos de las encuestas, omitiendo la primera línea
with open(archivo_datos, 'r', encoding='utf-8') as file:
    lineas = file.readlines()[1:] # Omite la primera línea del archivo de
datos

# Procesar cada línea del archivo de encuestas
for linea in lineas:
    linea = linea.strip().split(';') # Separar respuestas por ';'

    if len(linea) > 1: # Asegurar que hay más de una columna
        respuestas = linea[1:] # Omitir la primera columna

        # Crear un diccionario para la fila con todos los valores en 0
        fila = {col: 0 for col in encabezado}

        # Llenar con 1 las respuestas detectadas
        for respuesta in respuestas:
            if ':' in respuesta: # Verificar formato correcto (ej: 1:a,b)
                clave, valores = respuesta.split(':') # Separar número de
pregunta y opciones
                for valor in valores.split(','): # Separar opciones múltiples
                    columna = f"{clave}{valor.strip()}" # Construir el nombre
de la columna

                    if columna in fila:
                        fila[columna] = 1 # Asignar un 1 en la columna
correspondiente

        # Agregar la fila a la lista
        datos_procesados.append(fila)

# Crear el DataFrame con las columnas ordenadas según el encabezado
df = pd.DataFrame(datos_procesados, columns=encabezado).fillna(0).astype(int)

# **Añadir fila de sumatorio**
suma_filas = df.sum(axis=0) # Calcula la suma de cada columna
suma_filas_df = pd.DataFrame([suma_filas], columns=df.columns) # Convierte la
serie en DataFrame
suma_filas_df.index = ['Total'] # Etiqueta la fila

# Agregar la fila del sumatorio al DataFrame
df = pd.concat([df, suma_filas_df])

# Guardar en CSV con separador adecuado
df.to_csv(csv_resultados, index=True, sep=';', encoding='utf-8')

def txt_sum_grupo (csv_resultados): #Para juntar en un string el encabezado y el
valor sumatorio, de ahí va al txt de tránsito

    df = pd.read_csv(csv_resultados, sep=';', index_col=0, encoding='utf-8')

    # Obtener la fila de totales (última fila del DataFrame)
    fila_totales = df.iloc[-1] # Última fila

    # Crear una lista con los valores en formato "encabezado:total"
    resultado_final = [f"{col}:{int(fila_totales[col])}" for col in df.columns] #
Se fuerza a int para evitar decimales

    # Unir los valores en un string separado por ';'

```

```
resultado_str = ';'.join(resultado_final)

# Devolver los datos procesados
return resultado_str

def recopilatorio_datos (archivo_encabezado, archivo_datos_tratados,
archivo_recopilatorio): #para ir recopilando por grupos y cursos el tratamiento de
datos hecho anteriormente
    # Leer el encabezado del archivo
    with open(archivo_encabezado, 'r', encoding='utf-8') as file:
        encabezado = file.readline().strip().split(';') # Leer y separar
encabezado

    # Lista para almacenar los datos procesados
    datos_procesados = []

    # Leer los datos de los grupos
    with open(archivo_datos_tratados, 'r', encoding='utf-8') as file:
        lineas = file.readlines()

    # Procesar cada línea del archivo de resultados
    for linea in lineas:
        linea = linea.strip().split(';') # Separar elementos por ';'

        if len(linea) > 1: # Asegurar que hay datos en la línea
            nombre_grupo = linea[0] # Nombre del grupo
            respuestas = linea[1:] # Resto de valores

            # Crear un diccionario para la fila con valores en 0
            fila = {col: 0 for col in encabezado}
            fila["Grupo"] = nombre_grupo # Añadir nombre del grupo

            # Llenar con los valores del archivo
            for respuesta in respuestas:
                if ':' in respuesta: # Verificar formato correcto (ej: 1a:3)
                    clave, valor = respuesta.split(':') # Separar clave y valor
                    if clave in fila:
                        fila[clave] = int(valor) # Asignar el valor como entero

            # Agregar la fila a la lista
            datos_procesados.append(fila)

    # Crear el DataFrame asegurando que las columnas sigan el orden del encabezado
    columnas_ordenadas = ["Grupo"] + encabezado # Mantener el orden correcto
    df = pd.DataFrame(datos_procesados, columns=columnas_ordenadas).fillna(0)

    # Convertir solo las columnas numéricas a `int`
    columnas_numericas = [col for col in encabezado if col != "Grupo"] # Excluir
"Grupo"
    df[columnas_numericas] = df[columnas_numericas].astype(int)

    # **Añadir fila de sumatorio**
    suma_filas = df[columnas_numericas].sum(axis=0) # Calcula la suma de cada
columna numérica
    suma_filas_df = pd.DataFrame([suma_filas], columns=columnas_numericas) #
Convertir en DataFrame
    suma_filas_df.insert(0, "Grupo", "Total") # Añadir columna "Grupo" con el
nombre "Total"

    # Agregar la fila de sumatorio al DataFrame
    df = pd.concat([df, suma_filas_df], ignore_index=True)
```

```

# Guardar en CSV con separador adecuado
df.to_csv(archivo_recopilatorio, index=False, sep=';', encoding='utf-8')

def transformar_csv_a_txt(archivo_entrada, archivo_salida): #para generar los txt
    que se usan para que la LLM analice los resultados
    # Cargar el CSV
    df = pd.read_csv(archivo_entrada, sep=';', header=None)

    # Extraer los encabezados y la última fila
    encabezados = df.iloc[0, 1:].tolist() # Omitimos la primera columna
    valores = df.iloc[-1, 1:].tolist() # Última fila sin la primera columna

    # Estructura para almacenar los datos organizados
    datos_organizados = {}

    for encabezado, valor in zip(encabezados, valores):
        numero = ''.join(filter(str.isdigit, encabezado)) # Extrae el número
        letra = ''.join(filter(str.isalpha, encabezado)) # Extrae la letra

        if numero not in datos_organizados:
            datos_organizados[numero] = []
        datos_organizados[numero].append(f"{letra}:{valor}")

    # Escribir en el archivo de salida
    with open(archivo_salida, 'w', encoding='utf-8') as f:
        for numero, respuestas in datos_organizados.items():
            f.write(f"Pregunta {numero};" + ";" + ".join(respuestas) + ";\n")

    print(f" --> Archivo transformado guardado en: {archivo_salida}")

```

A5.4 Aux_graficas.py: código auxiliar para generar gráficas

```

import pandas as pd
import matplotlib.pyplot as plt
import os

def transformar_csv(archivo_entrada, archivo_salida):

    df = pd.read_csv(archivo_entrada, sep=';', header=None) #carga el CSV

    # Extracción los encabezados y la última fila:
    encabezados = df.iloc[0, 1:].tolist() # Omitimos la primera columna
    valores = df.iloc[-1, 1:].tolist() # Última fila sin la primera columna

    # Procesado los datos en una nueva estructura
    nueva_tabla = []
    for encabezado, valor in zip(encabezados, valores):
        numero = ''.join(filter(str.isdigit, encabezado)) # Extrae el número
        letra = ''.join(filter(str.isalpha, encabezado)) # Extrae la letra
        nueva_tabla.append([f'Pregunta {numero}', letra, valor])

    # Creación de DataFrame y exportado a CSV
    nuevo_df = pd.DataFrame(nueva_tabla, columns=['Pregunta', 'Letra', 'Valor'])
    nuevo_df.to_csv(archivo_salida, index=False, header=False, sep=';')
    print(f"Archivo transformado guardado en: {archivo_salida}")

def generar_graficas(csv_path, save_path):

    df = pd.read_csv(csv_path, delimiter=';', header=None) # Cargar el archivo CSV
    con el delimitador ';'
    df.columns = ['Pregunta', 'Opción', 'Valor'] # Asignación de nombres de
    columnas manualmente

```

```

df['Valor'] = pd.to_numeric(df['Valor'], errors='coerce') # convierte la
columna de valores a numérico
preguntas = df['Pregunta'].unique() #agrupa por la primera columna
for pregunta in preguntas:
    sub_df = df[df['Pregunta'] == pregunta]

    #GRÁFICO DE BARRAS:
    plt.figure(figsize=(8, 6))
    cmap = plt.get_cmap("Blues") #asignar colores según valores
    norm = plt.Normalize(sub_df['Valor'].min(), sub_df['Valor'].max())
    colores = [cmap(0.5 + 0.5 * norm(v)) for v in sub_df['Valor']]

    plt.bar(sub_df['Opción'], sub_df['Valor'], color=colores)
    plt.xlabel('Opción', fontsize=20, fontname='serif')
    plt.ylabel('Valor', fontsize=20, fontname='serif')
    plt.title(pregunta, fontsize=30, fontname='serif')
    plt.xticks(fontsize=16, fontname='serif')
    plt.yticks(fontsize=16, fontname='serif')
    plt.grid(axis='y', linestyle='--', alpha=0.7)

    save_file_bar = os.path.join(save_path, f"{pregunta}_barras.png") # Guarda
el gráfico de barras
    plt.savefig(save_file_bar, bbox_inches='tight')
    plt.close()

    #GRÁFICO CIRCULAR O DE PASTEL:
    sub_df_pie = sub_df[sub_df['Valor'] > 0] # Filtrar valores mayores que 0
para la gráfica de pastel
    cmap = plt.get_cmap('Dark2', len(sub_df_pie)) # genera una lista de
colores según la paleta 'Dark2'
    colores_usados = [cmap(i) for i in range(len(sub_df_pie))]

    plt.figure(figsize=(8, 6)) #crea la gráfica de pastel
    plt.pie(
        sub_df_pie['Valor'],
        labels=sub_df_pie['Opción'],
        autopct='%1.1f%%',
        colors=colores_usados,
        textprops={'fontsize': 20, 'fontname': 'serif'}, pctdistance=0.85
    )

    plt.title(pregunta, fontsize=30, fontname='serif')
    plt.tight_layout()

    save_file_pie = os.path.join(save_path, f"{pregunta}_pastel.png") #guarda
la imagen de pastel
    plt.savefig(save_file_pie, bbox_inches='tight')
    plt.close()

```

A5.5 Aux_informe.py: código auxiliar para generar los informes en PDF

```

import os
import re
from reportlab.lib.pagesizes import A4
from reportlab.pdfgen import canvas
from reportlab.lib.utils import ImageReader
from reportlab.platypus import Paragraph
from reportlab.lib.styles import getSampleStyleSheet

def generar_pdf(titulo, original_txt, ruta_dir_imagen, conclusiones_txt,
nombre_archivo, format):
    # Configurar página A4

```

```

c = canvas.Canvas(nombre_archivo, pagesize=A4)
width, height = A4 # Dimensiones de la página

# Conversión de cm a puntos (1 cm ≈ 28.35 pts)
cm = 28.35

# Configuración de tamaños y márgenes
img_height = 5 * cm # Altura fija de 5 cm para imágenes
margin = 1 * cm # 1 cm de margen lateral
top_margin_new_page = 2 * cm # Margen superior en nuevas páginas
section_spacing = 0.3 * cm # Espaciado de 0.3 cm entre secciones
bottom_margin = 2 * cm # Margen inferior de 2 cm
content_height = 1.5 * cm + section_spacing + img_height + section_spacing #
Altura del subtítulo + imagen + cuadro derecho

first_page = True # Variable para identificar si es la primera página

##### ENCABEZADO #####

# Configuración de imágenes y título
img_size = 3.5 * cm # 3.5 cm en puntos

# Dimensiones del recuadro del título
titulo_width = width - (2 * (img_size + margin + cm)) # Ancho entre las
imágenes
titulo_height = 3.5 * cm # Altura exacta de 3.5 cm

# Posición del recuadro del título (bajado 1 cm desde el borde superior)
titulo_x_min = (width - titulo_width) / 2 # Centrado horizontalmente
titulo_y_top = height - cm # Parte superior 1 cm debajo del borde superior
titulo_y_bottom = titulo_y_top - titulo_height # Altura del recuadro

# Configurar fuente y estilo del texto
styles = getSampleStyleSheet()
style = styles["Normal"]
style.fontName = "Helvetica-Bold"
style.fontSize = 18
style.alignment = 1 # Centrado
style.leading = 22 # Espaciado entre líneas

# Crear un párrafo con el texto ajustado al ancho del espacio disponible
texto_parrafo = Paragraph(titulo, style)

# Calcular la posición centrada del texto dentro del recuadro
texto_width, texto_height = texto_parrafo.wrap(titulo_width - 10,
titulo_height) # Ajustar al ancho
titulo_x_centro = titulo_x_min + (titulo_width - texto_width) / 2
titulo_y_centro = titulo_y_bottom + (titulo_height - texto_height) / 2

# Dibujar el texto en la posición calculada
texto_parrafo.drawOn(c, titulo_x_centro, titulo_y_centro)

# Posiciones de las imágenes (alineadas con la parte superior)
img_y_position = height - cm - img_size # Bajadas 1 cm junto con el título

# Dibujar imagen izquierda
imagen_izq = ImageReader("Recursos/ENM.jpg")
c.drawImage(imagen_izq, margin, img_y_position, width=img_size,
height=img_size)

# Dibujar imagen derecha
imagen_dcha = ImageReader("Recursos/CUD.jpg")

```

```

c.drawImage(imagen_dcha, width - img_size - margin, img_y_position,
width=img_size, height=img_size)

# Posición de la línea de separación (medio centímetro más arriba)
line_y_position = titulo_y_bottom - (0.5 * cm)

# Dibujar línea negra horizontal
c.setLineWidth(1)
c.line(margin, line_y_position, width - margin, line_y_position)

##### PREGUNTAS Y RESPUESTAS #####

# Posición inicial (1 cm debajo de la línea de separación)
y_position = height - (6 * cm) # Baja el cursor después del encabezado

# **Leer todas las líneas de los archivos de texto**
with open(original_txt, "r", encoding="utf-8") as file_info,
open(conclusiones_txt, "r",
encoding="utf-8") as file_datos:
    lineas_original = file_info.readlines() # Todas las líneas del archivo de
información
    lineas_conclusiones = file_datos.readlines() # Todas las líneas del
archivo de datos adicionales

# **Iterar sobre todas las líneas**
for i in range(len(lineas_original)):
    # Obtener datos de la línea actual de cada archivo
    linea_info = lineas_original[i].strip().split(";") if i <
len(lineas_original) else []
    datos_extra_texto = lineas_conclusiones[i].strip().split(':', 1)[-1] if i
< len(lineas_conclusiones) and ':' in lineas_conclusiones[i] else ""

    if len(linea_info) < 2:
        continue # Si no hay suficientes datos, saltar esta iteración

    # Extraer los primeros dos datos y añadir "Pregunta "
    subtítulo_texto = f"Pregunta {linea_info[0]} - {linea_info[1]}"

    # Extraer los datos restantes y realizar reemplazos
    texto_derecho = ";".join(linea_info[2:]) if len(linea_info) > 2 else "" #
Mantener ';' en el texto
    texto_derecho = texto_derecho.replace(";", "<br/>") # Reemplazar ';' por
saltos de línea en ReportLab
    texto_derecho = texto_derecho.replace(":", " ") # Reemplazar ':' por '
,

# **Verificar si hay espacio para el subtítulo, la imagen y el cuadro
derecho**
if y_position - content_height <= bottom_margin:
    c.showPage() # Crear nueva página
    first_page = False # Marcar que ya no es la primera página
    y_position = height - (top_margin_new_page if not first_page else
margin) # Aplicar margen superior solo en nuevas páginas

# **Dibujar el cuadro de subtítulo con bordes**
subtítulo_width = width - (2 * margin) # Ancho total menos márgenes
subtítulo_height = 1.5 * cm # Altura fija
c.rect(margin, y_position - subtítulo_height, subtítulo_width,
subtítulo_height, stroke=0, fill=0)

# Configurar estilos de texto
styles = getSampleStyleSheet()

```

```

style_subtitle = styles["Heading2"]
style_subtitle.fontName = "Helvetica-Bold"
style_subtitle.fontSize = 12
style_subtitle.leading = 16
style_subtitle.alignment = 4 # Texto justificado

# **Centrar texto en el cuadro del subtítulo**
subtitulo_parrafo = Paragraph(subtitulo_texto, style_subtitle)
subtitulo_parrafo.wrapOn(c, subtitulo_width - 10, subtitulo_height)
subtitulo_parrafo.drawOn(c, margin + 5, y_position - subtitulo_height / 2
- subtitulo_parrafo.height / 2)

y_position -= (subtitulo_height + section_spacing)

# **Dibujar la imagen debajo del subtítulo**

##### GRÁFICAS #####
#Selección de gráfica correspondiente:
graficas_desordenadas = sorted([os.path.splitext(archivo)[0] for _, _,
archivos in os.walk(ruta_dir_imagen) for archivo in archivos])

# Función para extraer el número de la pregunta
def extraer_numero(nombre):
    numeros = re.findall(r'\d+', nombre) # Encuentra todos los números en
el nombre
    return int(numeros[0]) if numeros else 0 # Convierte el primer número
encontrado a entero

# Ordenar los nombres extraídos por el número de la pregunta
graficas = sorted(graficas_desordenadas, key=extraer_numero)

if (format == 'a'): ruta_imagen = ruta_dir_imagen + "/" + graficas[2 * i
+ ".png" # elige solo los gráficos de barras
if (format == 'b'): ruta_imagen = ruta_dir_imagen + "/" + graficas[(2*i) +
1] + ".png" #elige solo los gráficos de pastel
print("La ruta de la gráfica es: " + ruta_imagen)

try:
    imagen = ImageReader(ruta_imagen)
    img_width, img_orig_height = imagen.getSize()
    aspect_ratio = img_width / img_orig_height
    img_width = img_height * aspect_ratio # Mantener la proporción

    c.drawImage(imagen, margin, y_position - img_height, width=img_width,
height=img_height)
except:
    print(f"Error cargando imagen: {ruta_imagen}")

# **Dibujar el cuadro de texto a la derecha de la imagen**
cuadro_x = margin + img_width + section_spacing # 1 cm a la derecha de la
imagen
cuadro_width = width - cuadro_x - margin # Ancho ajustado
c.rect(cuadro_x, y_position - img_height, cuadro_width, img_height,
stroke=0, fill=0)

# **Centrar texto en el cuadro a la derecha de la imagen**
style_text = styles["Normal"]
style_text.fontName = "Helvetica"
style_text.fontSize = 12
style_text.leading = 14
style_text.alignment = 4 # Texto justificado

texto_parrafo = Paragraph(texto_derecho, style_text)
    
```

```
texto_parrafo.wrapOn(c, cuadro_width - 10, img_height)
texto_parrafo.drawOn(c, cuadro_x + 5, y_position - img_height / 2 -
texto_parrafo.height / 2)

y_position -= (img_height + section_spacing)

# **Calcular altura del cuadro de texto final**
datos_width = width - (2 * margin) # Ancho completo con márgenes
datos_parrafo = Paragraph(datos_extra_texto, styles["Normal"])
datos_width_text, datos_height_text = datos_parrafo.wrap(datos_width - 10,
height - y_position)

# **Si el cuadro de texto final no cabe, moverlo a la siguiente página**
if y_position - datos_height_text <= bottom_margin:
    c.showPage() # Crear nueva página
    first_page = False # Marcar que ya no es la primera página
    y_position = height - (top_margin_new_page if not first_page else
margin) # Aplicar margen superior solo en nuevas páginas

# Dibujar el recuadro con bordes
c.rect(margin, y_position - datos_height_text, datos_width,
datos_height_text, stroke=0, fill=0)

# **Centrar texto en el cuadro de Datos.txt**
datos_parrafo.drawOn(c, margin + 5, y_position - datos_height_text / 2 -
datos_parrafo.height / 2)

y_position -= (datos_height_text + section_spacing)

# Dibujar el subtítulo 'Conclusiones del análisis:'
conclusiones_titulo = "Conclusiones del análisis:"
subtitulo_width = width - (2 * margin)
subtitulo_height = 1.5 * cm

#####

# Dibujar el cuadro con la última línea de 'conclusiones_txt'
ultima_conclusion = lineas_conclusiones[-1].strip().split('; ', 1)[-1] if
lineas_conclusiones and ';' in lineas_conclusiones[-1] else ""

datos_parrafo = Paragraph(ultima_conclusion, styles["Normal"])

datos_width = width - (2 * margin)
# Ajustar la altura del cuadro al contenido del texto
datos_parrafo_width, datos_parrafo_height = datos_parrafo.wrap(datos_width -
10, height - y_position)

#####
# **Verificar si hay espacio para el subtítulo, la imagen y el cuadro
derecho**
if y_position - datos_parrafo_height - subtitulo_height <= bottom_margin:
    c.showPage() # Crear nueva página
    first_page = False # Marcar que ya no es la primera página
    y_position = height - (top_margin_new_page if not first_page else margin)
# Aplicar margen superior solo en nuevas páginas

#####

c.rect(margin, y_position - subtitulo_height, subtitulo_width,
subtitulo_height, stroke=0, fill=0)
style_subtitle = styles["Normal"]
```

```

style_subtitle.fontName = "Helvetica-Bold"
style_subtitle.fontSize = 12
style_subtitle.leading = 16
style_subtitle.alignment = 4
subtitulo_parrafo = Paragraph(conclusiones_titulo, style_subtitle)
subtitulo_parrafo.wrapOn(c, subtitulo_width - 10, subtitulo_height)
subtitulo_parrafo.drawOn(c, margin + 5, y_position - subtitulo_height / 2 -
subtitulo_parrafo.height / 2)

y_position -= (subtitulo_height + section_spacing)

#####

c.rect(margin, y_position - datos_parrafo_height, datos_width,
datos_parrafo_height, stroke=0, fill=0)
#datos_parrafo = Paragraph(ultima_conclusion, styles["Normal"])
datos_parrafo.wrapOn(c, datos_width - 10, 3 * cm)
datos_parrafo.drawOn(c, margin + 5, y_position - datos_parrafo_height / 2 -
datos_parrafo.height / 2)

y_position -= (2 * cm + section_spacing)

# Espacio de 1 cm antes del final
y_position -= (1 * cm)

# **Guardar PDF**
c.save()

```

ANEXO VI: CÓDIGOS DE IMPLEMENTACIÓN DE LLM

A6.1 LLM_DeepSeek.py: código de implementación del modelo DeepSeek-R1 7B

```
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM

model_id = "deepseek-ai/DeepSeek-R1-Distill-Qwen-7B"

tokenizer = AutoTokenizer.from_pretrained(model_id, trust_remote_code=True)
model = AutoModelForCausalLM.from_pretrained(
    model_id,
    torch_dtype=torch.float16,
    device_map="auto",          # Prioriza uso de GPU
    trust_remote_code=True
)

model.eval()
print("\n\nDeepSeek-R1 cargado correctamente\n")

def analisis_LLM(ruta_prompt):

    with open(ruta_prompt, "r", encoding="utf-8") as file:
        prompt = file.read()

    inputs = tokenizer(
        prompt,
        return_tensors="pt",
        padding=True,
        truncation=True
    ).to(model.device)

    with torch.no_grad():
        outputs = model.generate(
            inputs["input_ids"],
            attention_mask=inputs["attention_mask"],
            max_new_tokens=1000, # Limita los tokens utilizados
            temperature=0.2,    # Controla creatividad (mayor, más creatividad) para
                                # evitar alucinaciones
            top_p=0.5,          # Reduce diversidad, es más directo con su respuesta
            pad_token_id=tokenizer.eos_token_id,
            eos_token_id=tokenizer.eos_token_id, # Forzar fin de generación
        )

    response = tokenizer.decode(outputs[0], skip_special_tokens=True)
    conclusion = response.split("</think>", 1)[-1].strip() # Tomamos lo que está
    # después y eliminamos espacios

    return conclusion
```

A6.2 LLM_Llama3.py: código de implementación del modelo Llama-3 de Meta

```
from huggingface_hub import login
login(token="****") #Introducir código access token de huggingface
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM, BitsAndBytesConfig,
pipeline

print("CUDA disponible:", torch.cuda.is_available())
print("Número de GPUs:", torch.cuda.device_count())
print("Nombre de la GPU:", torch.cuda.get_device_name(0) if
torch.cuda.is_available() else "No hay GPU")
```

```

opcion = input("¿Desea usar cuantificación?[y/n]: ").lower()
while (opcion != "y" and opcion != "n"): opcion = str(input("¿Desea usar
cuantificación?[y/n]: "))

model_id = "meta-llama/Meta-Llama-3-8B"

if opcion == "n":
    tokenizer = AutoTokenizer.from_pretrained(model_id)
    tokenizer.pad_token = tokenizer.eos_token

    model = AutoModelForCausalLM.from_pretrained(
        model_id,
        device_map="cuda:0", # Forzar ejecución en GPU
        torch_dtype=torch.float32 # Evitar uso de 4-bit o 8-bit
    )

if opcion == "y":
    bnb_config = BitsAndBytesConfig(
        load_in_4bit=True,
        bnb_4bit_use_double_quant=True,
        bnb_4bit_quant_type="nf4",
        bnb_4bit_compute_dtype=torch.bfloat16
    )

    tokenizer = AutoTokenizer.from_pretrained(model_id)
    tokenizer.pad_token = tokenizer.eos_token
    model = AutoModelForCausalLM.from_pretrained(
        model_id,
        device_map="auto",
        quantization_config=bnb_config
    )

text_generator = pipeline(
    "text-generation",
    model=model,
    tokenizer=tokenizer,
    max_new_tokens=128,
)

print("\n\nLlama3 de Meta cargado correctamente\n")

def analisis_LLM (ruta_prompt):
    with open(ruta_prompt, "r", encoding="utf-8") as file:
        prompt = file.read()

    response = text_generator(prompt)
    respuesta = response[0]['generated_text']
    return respuesta

```