



UNIVERSITAT  
POLITÈCNICA  
DE VALÈNCIA



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Detectando desinformación a través de técnicas de  
argumentación computacional y grandes modelos de  
lenguaje

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Gutiérrez Mandingorra, Ana

Tutor/a: Heras Barberá, Stella María

Cotutor/a: Palanca Cámara, Javier

CURSO ACADÉMICO: 2023/2024



# Resum

En l'actualitat, la difusió de la desinformació planteja un desafiament de gran envergadura per a la societat. Els ciutadans es troben immersos en un context digital complex i saturat de dades que els dificulta discernir críticament entre informació veraç i falsa. La tasca és complexa ja que la desinformació es presenta sovint en forma de manipulacions retòriques, fal·làcies lògiques o distorsions de la veritat. Aquest projecte té com a objectiu desenvolupar un sistema per a la detecció de desinformació basat en tècniques d'argumentació computacional i grans models de llenguatge, que fomenti el pensament crític i l'alfabetització mediàtica en la societat.

L'eina web implementada analitza els patrons de raonament humà utilitzats al argumentar, classificant els arguments en esquemes argumentatius definits per la teoria de l'argumentació. Després d'identificar l'esquema argumentatiu, el sistema examina detalladament el raonament exposat en l'argument i fa servir un conjunt de preguntes crítiques per a qüestionar la seua validesa. Mitjançant un gran model de llenguatge, potenciat amb contextualització externa de diverses fonts d'informació, es guia al sistema en el procés d'avaluació de veracitat de l'argument. La resposta final inclou una justificació tant qualitativa com quantitativa del nivell de veracitat, proporcionant els enllaços i referències a les fonts d'informació utilitzades en l'avaluació.

Per al desenvolupament d'aquest sistema, s'han utilitzat diverses tecnologies, incloent-hi el *framework* Rasa per a la implementació del sistema classificador i un gran model de llenguatge de LLAMA connectat a diferents APIs per a l'avaluació de veracitat. A més, s'han dut a terme múltiples experiments per a seleccionar les aproximacions que demostren un millor rendiment. Finalment, s'ha avaluat el funcionament final del sistema tant quantitativament com qualitativament.

**Paraules clau:** argumentació computacional, grans models de llenguatge (LLM), processament de llenguatge natural (PLN), RASA, LLAMA, sistema classificador, sistema generador, eina web

---

# Resumen

En la actualidad, la difusión de la desinformación plantea un desafío de gran envergadura para la sociedad. Los ciudadanos se encuentran inmersos en un contexto digital complejo y saturado de datos que les dificulta discernir críticamente entre información veraz y falsa. La tarea es compleja ya que la desinformación se presenta a menudo en forma de manipulaciones retóricas, falacias lógicas o distorsiones de la verdad. Este proyecto tiene como objetivo desarrollar un sistema para la detección de desinformación basado en técnicas de argumentación computacional y grandes modelos de lenguaje, que fomente el pensamiento crítico y la alfabetización mediática en la sociedad.

La herramienta web implementada analiza los patrones de razonamiento humano utilizados al argumentar, clasificando los argumentos en esquemas argumentativos definidos por la teoría de la argumentación. Tras identificar el esquema argumentativo, el sistema examina detalladamente el razonamiento expuesto en el argumento y utiliza un conjunto de preguntas críticas para cuestionar su validez. Mediante un gran modelo de lenguaje, potenciado con contextualización externa de diversas fuentes de información, se guía al sistema en el proceso de evaluación de veracidad del argumento. La respuesta final incluye una justificación tanto cualitativa como cuantitativa del nivel de veracidad, proporcionando los enlaces y referencias a las fuentes de información utilizadas en la evaluación.

Para el desarrollo de este sistema, se han utilizado diversas tecnologías, incluyendo el *framework* Rasa para la implementación del sistema clasificador y un gran modelo de lenguaje de LLAMA conectado a diferentes APIs para la evaluación de veracidad. Además, se han llevado a cabo múltiples experimentos para seleccionar las aproximaciones que demostraban un mejor rendimiento. Finalmente, se ha evaluado el funcionamiento final del sistema tanto cuantitativamente como cualitativamente.

**Palabras clave:** argumentación computacional, grandes modelos de lenguaje (LLM), procesamiento de lenguaje natural (PLN), RASA, LLAMA, sistema clasificador, sistema generador de texto, herramienta web

---

# Abstract

Nowadays, the spread of disinformation poses a major challenge for society. Citizens find themselves immersed in a complex and data-saturated digital context that hinders their ability to critically discern between true and false information. The task is complex because disinformation often appears in the form of rhetorical manipulations, logical fallacies, or distortions of the truth. This project aims to develop a system for detecting disinformation based on computational argumentation techniques and large language models, which promotes critical thinking and media literacy in society.

The implemented web tool analyzes the patterns of human reasoning used in argumentation, classifying the arguments into argumentative schemes defined by argumentation theory. After identifying the argumentative scheme, the system thoroughly examines the reasoning presented in the argument and uses a set of critical questions to question its validity. Using a large language model, enhanced with external contextualization from various information sources, the system is guided in the process of evaluating the truthfulness of the argument. The final response includes both a qualitative and quantitative justification of the level of truthfulness, providing links and references to the information sources used in the evaluation.

For the development of this system, various technologies have been used, including the Rasa framework for implementing the classifier system and a large language model from LLAMA connected to different APIs for truthfulness evaluation. Additionally, multiple experiments were conducted to select the approaches that demonstrated the best performance. Finally, the final functioning of the system was evaluated both quantitatively and qualitatively.

**Key words:** computational argumentation, large language models (LLM), natural language processing (NLP), RASA, LLAMA, classifier system, text generator system, web tool

---



# Índice general

---

<b>Índice general</b>	VII
<b>Índice de figuras</b>	IX
<b>Índice de tablas</b>	X
<hr/>	
<b>1 Introducción</b>	<b>1</b>
1.1 Motivación . . . . .	1
1.2 Objetivos . . . . .	1
1.3 Impacto Esperado . . . . .	2
1.4 Metodología . . . . .	2
1.5 Estructura de la memoria . . . . .	3
1.6 Colaboraciones . . . . .	4
<b>2 Estado del arte</b>	<b>5</b>
2.1 Teoría de la argumentación . . . . .	5
2.2 La sociedad de la desinformación . . . . .	7
2.3 Procesamiento de lenguaje natural . . . . .	8
2.4 Argument Mining . . . . .	11
2.4.1 Modelos de lenguaje tradicionales . . . . .	12
2.4.2 Grandes modelos de lenguaje basados en Transformers . . . . .	13
2.4.3 Chatbots basados en grandes modelos de lenguaje . . . . .	14
2.5 Crítica al estado del arte . . . . .	15
2.5.1 Limitaciones de los LLMs . . . . .	15
2.5.2 Sistema Multiproceso con conexión a Fuentes Externas . . . . .	16
2.6 Propuesta . . . . .	17
<b>3 Análisis del Problema</b>	<b>19</b>
3.1 Especificación de requisitos . . . . .	20
<b>4 Diseño de la solución</b>	<b>23</b>
4.1 Arquitectura del sistema . . . . .	23
4.2 Diseño detallado . . . . .	24
4.2.1 Módulo 1: Sistema clasificador en esquemas argumentativos . . . . .	25
4.2.2 Módulo 2: Sistema generador y evaluador del nivel de veracidad . . . . .	26
4.2.3 Interfaz web del sistema . . . . .	28
4.3 Tecnología utilizada . . . . .	28
4.3.1 Rasa . . . . .	28
4.3.2 LLAMA . . . . .	29
4.3.3 LangChain . . . . .	29
4.3.4 Otras librerías de Python . . . . .	30
4.3.5 Entornos y sistema operativo . . . . .	31
<b>5 Desarrollo de la solución propuesta</b>	<b>33</b>
5.1 Módulo 1: Sistema clasificador en esquemas argumentativos . . . . .	33
5.1.1 Dataset utilizado . . . . .	33
5.1.2 Entrenamiento con Rasa . . . . .	35
5.2 Módulo 2: Sistema generador y evaluador del nivel de veracidad . . . . .	38

5.2.1	Configuración y uso de LLAMA	38
5.2.2	Recuperación de información	40
5.3	Interconexión de los módulos	41
5.4	Implementación de la herramienta web	42
<b>6</b>	<b>Experimentación y resultados</b>	<b>47</b>
6.1	Módulo 1: Sistema clasificador en esquemas argumentativos	47
6.1.1	Evaluación del experimento unicapa	49
6.1.2	Evaluación del experimento bicapa	50
6.1.3	Conclusión de los experimentos	53
6.2	Módulo 2: Sistema generador y evaluador del nivel de veracidad	55
6.2.1	Experimento LLAMA 2 con la librería de LangChain	55
6.2.2	Experimento LLAMA 3 con la API REST de PoliGPT	57
6.2.3	Conclusión de los experimentos	58
6.3	Evaluación general del sistema	58
<b>7</b>	<b>Conclusiones</b>	<b>63</b>
7.1	Cumplimiento de los objetivos	63
7.2	Conocimientos adquiridos y dificultades encontradas	64
7.3	Relación del trabajo desarrollado con los estudios cursados	64
7.4	Trabajos futuros	66
	<b>Bibliografía</b>	<b>67</b>
<hr/>		
Apéndices		
<b>A</b>	<b>Objetivos de Desarrollo Sostenible</b>	<b>71</b>
<b>B</b>	<b>GLOSARIO</b>	<b>73</b>



# Índice de figuras

---

2.1	Diagrama argumental	6
2.2	Ejemplos de Redes Neuronales Recurrentes	10
2.3	Ejemplo de Red Neuronal Convolutiva	10
2.4	Ejemplo de la arquitectura Transformer	11
2.5	Técnica de priming y prompting	15
2.6	Ejemplo de limitación de un LLM	17
3.1	Valoración del nivel de preocupación sobre la propagación de desinformación	19
3.2	Valoración de los ámbitos más afectados por la desinformación	20
4.1	Esquema del diseño de la arquitectura del sistema	23
4.2	Esquema detallado del diseño de la arquitectura del sistema	24
4.3	Esquema detallado del diseño del Módulo 1	25
4.4	Esquema detallado del diseño del Módulo 2	27
4.5	Mockup diseño del sitio web	28
5.1	Distribución del dataset utilizado	34
5.2	Fragmento ejemplo de archivo YAML	35
5.3	Ejemplo de intención definida en Rasa	35
5.4	Intenciones especificadas en el clasificador de la capa 1 del sistema clasificador de Rasa	36
5.5	Ejemplos especificados en Rasa para cada intención del clasificador de la capa 1	37
5.6	Flujo de las solicitudes realizadas al modelo	40
5.7	Introducción a la herramienta web	43
5.8	Introducción del argumento a evaluar en la herramienta web	43
5.9	Clasificación en la herramienta web del argumento en un esquema argumentativo	44
5.10	Justificación cualitativa del nivel de veracidad en la herramienta web	45
5.11	Justificación cuantitativa del nivel de veracidad en la herramienta web	45
5.12	Fuentes utilizadas en la justificación del nivel de veracidad en la herramienta web	45
6.1	Experimentos módulo 1	47
6.2	Matriz de confusión experimento unicapa	51
6.3	Matriz de confusión capa 1 del experimento bicapa	52
6.4	Matriz de confusión capa 2 grupo 1 del experimento bicapa	53
6.5	Matriz de confusión capa 2 grupo 2 del experimento bicapa	53
6.6	Matriz de confusión capa 2 grupo 3 del experimento bicapa	53
6.7	Comparación del rendimiento global de los 2 experimentos	54
6.8	Ejemplo del funcionamiento de un agente en LangChain	56
6.9	Comparativa de la versión secuencial y paralela del sistema	57
6.10	Valoración de la justificación cuantitativa del sistema	59

6.11	Valoración de la justificación cualitativa del sistema . . . . .	59
6.12	Valoración de la adecuación de las fuentes que utiliza el sistema . . . . .	59
6.13	Valoración general del sistema . . . . .	60
6.14	Comparativa de las valoraciones de cada aspecto del sistema . . . . .	60
6.15	Valoración de la utilidad del sistema . . . . .	61

## Índice de tablas

---

2.1	Familias de Esquemas Argumentativos . . . . .	6
2.2	Esquemas argumentativo de Autoridad . . . . .	6
2.3	Esquemas argumentativo de Ejemplo . . . . .	7
2.4	Esquemas argumentativo de Desperdicio . . . . .	7
2.5	Comparativa de arquitecturas y modelos destacados . . . . .	12
3.1	Especificación de Requisitos del Sistema de Detección de Desinformación	21
6.1	Métricas experimento unicapa . . . . .	50
6.2	Métricas capa 1 del experimento bicapa . . . . .	52
6.3	Métricas capa 2 del experimento bicapa . . . . .	53

# AGRADECIMIENTOS

---

En primer lugar, me gustaría agradecer al Departamento de Sistemas Informáticos y Computación de la Universitat Politècnica de València por brindarme la oportunidad de trabajar en este apasionante proyecto a través de la beca de colaboración que me fue concedida.

Gracias a esta beca, he podido colaborar y trabajar directamente con miembros del Grupo de Tecnología Informática Inteligencia Artificial (GTI-IA) del *Valencian Research Institute for Artificial Intelligence* (VRAIN). En particular, quisiera agradecer a mis tutores, la Dra. Stella María Heras Barberá y el Dr. Javier Palanca Cámara, por su inestimable guía, consejos y resolución de dudas a lo largo de este proyecto. Su experiencia y apoyo no solo fueron esenciales para la realización de este proyecto, sino que también me brindaron una valiosa oportunidad de adquirir nuevos conocimientos y habilidades.

Quisiera también extender mi agradecimiento a las 80 personas que participaron en la encuesta realizada para evaluar el sistema final. Su colaboración fue fundamental para evaluar los resultados y llevar a buen término este proyecto.

Por último, quiero agradecer a mi familia, especialmente a mis padres, por su incondicional apoyo, ánimo y fortaleza durante los meses dedicados a este proyecto y a lo largo de toda mi etapa universitaria.



---

---

# CAPÍTULO 1

## Introducción

---

### 1.1 Motivación

---

En el panorama actual, caracterizado por la digitalización de la información, la propagación de la desinformación representa un desafío crítico para la sociedad. El gran flujo de información generado por las redes sociales, los medios de comunicación y las plataformas en línea hace que cada vez sea más difícil discernir entre información veraz y desinformación.

Por tanto, reconociendo que la desinformación a menudo se presenta en forma de falacias lógicas, manipulaciones retóricas o distorsiones de la verdad, es fundamental para la sociedad combatir la propagación de declaraciones falsas o contenido engañoso con intención manipulativa o persuasiva. En este contexto, se ha considerado necesario investigar cómo se puede aplicar el ámbito de la Inteligencia Artificial con el fin de contrarrestar el fenómeno de la desinformación. Para ello, suelen ser típicas las soluciones que utilizan técnicas de procesamiento de lenguaje natural y modelos de aprendizaje automático.

Sin embargo, para lograr una mayor eficacia en la lucha contra la desinformación, es esencial combinar estas tecnologías con técnicas propias del campo de la argumentación computacional. La argumentación computacional añade una capa lógica que refuerza las tecnologías tradicionales, permitiendo generar una línea de razonamiento más segura y veraz. Esta combinación de tecnologías tiene el potencial de desarrollar herramientas que no solo identifiquen la desinformación, sino que también capaciten a los ciudadanos para cuestionar y evaluar la veracidad de los hechos de manera crítica.

El presente trabajo aborda este desafío mediante el desarrollo de un sistema de detección de desinformación basado en la aplicación de técnicas argumentativas y el uso de un gran modelo de lenguaje potenciado con contextualización externa. El sistema analiza los diferentes patrones de razonamiento humano presentes en los argumentos, clasificándolos en esquemas argumentativos determinados por la teoría de la argumentación. A partir de la identificación de estos patrones, se guía al sistema en el proceso de evaluación de la veracidad de estos argumentos, proporcionando una justificación final sobre su nivel de veracidad que fomente el pensamiento crítico de los usuarios.

### 1.2 Objetivos

---

El objetivo principal de este proyecto es el desarrollo de una herramienta capaz de detectar desinformación en argumentos expuestos por la sociedad en formato textual que pueden presentarse a modo de falacias lógicas, manipulaciones retóricas o distorsiones

de la verdad. Para alcanzar este objetivo general, es necesario cumplir con los siguientes objetivos específicos:

- **OE1:** Revisar y analizar el estado del arte del ámbito de la detección de desinformación y argumentación computacional para conocer los trabajos e investigaciones que se han llevado a cabo y familiarizarse con la temática.
- **OE2:** Implementar un módulo clasificador de esquemas argumentativos que incorpore conceptos y técnicas propias de la argumentación computacional en el sistema final. Este módulo deberá clasificar correctamente los argumentos en sus correspondientes esquemas argumentativos.
- **OE3:** Implementar un módulo evaluador de la veracidad de los argumentos utilizando un modelo de lenguaje grande guiado por la identificación de patrones lógicos. Este módulo deberá presentar la evaluación del nivel de veracidad del argumento proporcionando una justificación tanto cualitativa como cuantitativa.
- **OE4:** Proporcionar justificaciones a las decisiones tomadas por el modelo, potenciando el LLM con contextualización externa mediante una fase de recuperación de información en diversas fuentes. Los enlaces a las fuentes de información externa deberán ser mostrados al usuario, mejorando la transparencia e información suministrada por el sistema.
- **OE5:** Desarrollar una interfaz web intuitiva que facilite la interacción del usuario con la herramienta, permitiéndole introducir argumentos y recibir las evaluaciones de manera clara.
- **OE6:** Mejorar y optimizar el rendimiento de la herramienta final mediante la implementación de un sistema con paralelización de tareas utilizando técnicas de programación concurrente.
- **OE7:** Evaluar la efectividad del sistema a través del análisis de resultados obtenidos a partir de evaluaciones cuantitativas y cualitativas.

### 1.3 Impacto Esperado

---

La herramienta a desarrollar para la detección de desinformación tiene un impacto económico y social significativo, con implicaciones positivas en diversos sectores de la sociedad alineándose con varios Objetivos de Desarrollo Sostenible (ODS) de las Naciones Unidas (ver Anexo A). Al proporcionar evaluaciones claras y justificadas sobre la veracidad de los argumentos, la herramienta permite a los usuarios tomar decisiones informadas, desarrollar un pensamiento crítico y en consecuencia reducir los costes de identificar y desmentir esta información falsa. Por tanto, se mejora la alfabetización mediática, fomentando una sociedad más informada y responsable en el debate público.

### 1.4 Metodología

---

Durante el desarrollo de este trabajo, se ha adoptado una metodología estructurada y ágil para asegurar el progreso constante y garantizar la calidad del proyecto.

Se han realizado reuniones semanales con los tutores del TFG, tanto de manera presencial como online a través de Microsoft Teams. En cada reunión, se discutían los avances logrados, se realizaban las correcciones oportunas y se establecían los siguientes objetivos en intervalos de una o dos semanas. Esta dinámica ha permitido un seguimiento

continuo del proyecto, facilitando la resolución de problemas de manera oportuna y la incorporación de mejoras continuas basadas en los comentarios recibidos durante las reuniones semanales.

Para la planificación y gestión de tareas se ha utilizado Notion, una herramienta fundamental en este proyecto en la que se registraban todas las tareas, avances, actas de reunión y seguimientos durante las diferentes etapas del proyecto. Notion ha permitido centralizar toda la información relevante en un solo lugar, facilitando la organización y el acceso a los datos necesarios en las diferentes etapas del proyecto.

Durante la fase de investigación, se creó un espacio a modo de biblioteca en Notion para almacenar y organizar toda la información recopilada: notas detalladas, referencias, artículos académicos, resúmenes, esquemas, recursos web, vídeos, tutoriales, etc. En la etapa de diseño y desarrollo, se siguieron anotando los avances en Notion y, una vez finalizada la implementación de la herramienta, se creó un repositorio de código en GitHub donde se alojó y documentó todo el código y el sistema de archivos creado. El código del proyecto desarrollado se encuentra disponible bajo demanda en el siguiente repositorio<sup>1</sup>. En la fase de experimentación y pruebas, los resultados se registraron tanto en Notion como en hojas de Excel, asegurando la precisión e integridad de los datos y permitiendo un análisis detallado de los resultados obtenidos.

Una vez finalizado el proyecto, se procedió a redactar su correspondiente memoria, documentando detalladamente todo el proceso seguido, los conocimientos aplicados, los problemas surgidos, las soluciones propuestas y los resultados obtenidos. Para la redacción de la memoria se utilizó el entorno Overleaf, una plataforma colaborativa basada en LaTeX, que facilitó la escritura, revisión y formato del documento. A lo largo de esta fase, se realizaron revisiones continuas de los diferentes capítulos de la memoria con los tutores del TFG, asegurando que cada sección del documento cumpliera con los estándares académicos y reflejara con precisión el trabajo realizado.

## 1.5 Estructura de la memoria

---

La memoria se estructura en ocho capítulos con anexos finales. En el capítulo 1 se presenta una introducción del proyecto desarrollado. En el capítulo 2 se explica el estado del arte en el ámbito del procesamiento de lenguaje natural, la argumentación computacional y la detección de desinformación. En el capítulo 3 se realiza un análisis del problema y se especifican los requisitos. El capítulo 4 presenta el diseño general del sistema detector de desinformación, así como el diseño específico de los módulos que lo componen y las tecnologías que se han utilizado a lo largo del proyecto. El capítulo 5 detalla el desarrollo de cada módulo del sistema, su interconexión y la implementación final de la interfaz web. En el capítulo 6 se presentan los experimentos realizados en cada módulo, junto con sus resultados, y se comparan diferentes aproximaciones con el objetivo de determinar la más adecuada para la versión final del sistema. Finalmente, en el capítulo 7, se exponen las conclusiones del trabajo. Además, se incluye en el Anexo A los Objetivos de Desarrollo Sostenible (ODS) relacionados con el proyecto y en el Anexo B un glosario con los términos principales utilizados.

---

<sup>1</sup>Código del proyecto en repositorio privado de: <https://github.com/anagutierr>

## 1.6 Colaboraciones

---

El desarrollo de este trabajo final de grado se enmarca en una beca de colaboración del Ministerio de Educación y Formación Profesional realizada durante el curso 2023-2024 (*Número del Proyecto: 2023/32/00007*). El objetivo de esta beca era desarrollar un modelo basado en técnicas de argumentación computacional e Inteligencia Artificial para la detección y análisis de intentos de desinformación.

La beca de colaboración ha sido propuesta por el Departamento de Sistemas Informáticos y Computación (DSIC) de la UPV. Concretamente los responsables de esta beca han sido la Dra. Stella María Heras Barberá, investigadora en el Grupo de tecnología informática Inteligencia Artificial (GTI-IA)<sup>2</sup>, y el Dr. Vicente Juan Botti Navarro, director del Instituto de Ingeniería Artificial de Valencia (VRAIN)<sup>3</sup>. Así que, durante el desarrollo del proyecto presentado en el TFG se ha colaborado y trabajado directamente con miembros del GTI-IA del VRAIN, como son la Dra. Stella María Heras Barberá y el Dr. Javier Palanca Cámara, tutora y cotutor de este TFG.

Por otra parte, en el desarrollo de este proyecto se ha utilizado una base de datos etiquetada sobre argumentación computacional que fue constituida a través de una colaboración del VRAIN y el *Centre for Argument Technology* (ARG-tech)<sup>4</sup> de la Universidad de Dundee (UK). En el artículo [Ruiz-Dolz et al., 2024] se recoge esta colaboración y se presenta el *dataset* utilizado en este proyecto. Consiste en un corpus multilingüe de esquemas de argumentación en lenguaje natural generados automáticamente, llamado NLAS-MULTI<sup>5</sup>.

---

<sup>2</sup><https://gti-ia.webs.upv.es/>

<sup>3</sup><https://vrain.upv.es/>

<sup>4</sup><https://www.arg.tech/>

<sup>5</sup><https://zenodo.org/records/8364002>



---

---

## CAPÍTULO 2

# Estado del arte

---

Este capítulo muestra la investigación llevada a cabo sobre el estado del arte en la detección de desinformación, abordando el objetivo OE1 de la sección 1.2. Además, proporciona un marco teórico introductorio para comprender el contexto y los conceptos fundamentales de la argumentación computacional.

### 2.1 Teoría de la argumentación

---

Desde la antigüedad, filósofos griegos como Sócrates, Platón y Aristóteles exploraron el campo de la retórica, los principios de la argumentación y la persuasión. Su objetivo radicaba en entender cómo los humanos son capaces de razonar mediante la formulación de argumentos e incluso de cometer errores de razonamiento. La teoría de la argumentación es el campo que estudia cómo se construyen, evalúan y utilizan los argumentos en diferentes contextos y disciplinas.

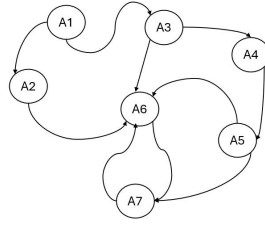
Tal y como se explica en [Walton, 2005], un argumento es un conjunto de afirmaciones compuesto por tres partes: unas premisas, unas conclusiones y una inferencia de las premisas a las conclusiones. Los argumentos se relacionan entre ellos, pudiendo estar un argumento respaldado o atacado por otros. Según la forma de razonamiento seguida, hay dos tipos de argumentos: los deductivos, donde la conclusión se deriva necesariamente de las premisas; y los inductivos, que infieren conclusiones a partir de observaciones para generalizar reglas.

Según el enfoque elegido, destacan dos modelos clave en el análisis de argumentos:

- **El modelo monológico (argumentación estructurada):** el cual se centra en las relaciones de las diferentes partes de un argumento.
- **El modelo dialógico (argumentación abstracta):** que representa cada argumento como un elemento atómico y se centra en cómo se relacionan los argumentos entre ellos.

Los diagramas o mapas argumentales son de gran utilidad para representar de manera visual este tipo de relaciones. En la Figura 2.1 se muestra un ejemplo de diagrama argumental formado por varios argumentos.

Para evaluar la validez y la lógica de un argumento en nuestro sistema, se utilizan los esquemas argumentativos. Estos son patrones de razonamiento abstractos que seguimos los humanos. Están formados por un conjunto de premisas, una conclusión y un conjunto de cuestiones críticas que sirven para cuestionar la validez del argumento. En [Walton et al., 2008] se propone un sistema general para la agrupación de los diferentes



**Figura 2.1:** Ejemplo de diagrama/mapa argumental en el que se observan las relaciones de ataque (aristas) entre diferentes argumentos (nodos)

esquemas argumentativos en tres grandes grupos o familias. En la Tabla 2.1 se muestra esta clasificación general y en las Tablas 2.2, 2.3 y 2.4 se muestran 3 ejemplos de esquema argumentativos, concretamente, sobre el argumento de autoridad, el de ejemplo y el de desperdicio con sus respectivas cuestiones críticas.

Argumentos basados en fuentes	Argumentos que aplican reglas a casos particulares	Argumentos de razonamiento
<ul style="list-style-type: none"> <li>• Argumento de Posición de Saber</li> <li>• Argumento de Testimonio</li> <li>• Argumento de Opinión de Expertos</li> <li>• Argumento de Ignorancia</li> <li>• Compromiso Inconsistente</li> <li>• Direct ad Hominem</li> <li>• Argumento de Opinión Popular</li> <li>• Argumento de Práctica Popular</li> </ul>	<ul style="list-style-type: none"> <li>• Argumento de Ejemplo</li> <li>• Argumento de Analogía</li> <li>• Argumento de Precedente</li> <li>• Argumento de la Regla Establecida</li> <li>• Argumento de Clasificación Verbal</li> </ul>	<ul style="list-style-type: none"> <li>• Argumento de Malgasto</li> <li>• Argumento de Costos Hundidos</li> <li>• Argumento de Amenaza</li> <li>• Argumento de Signo</li> <li>• Argumento de Causa a Efecto</li> </ul>

**Tabla 2.1:** Clasificación general de esquemas argumentativos en familias según [Walton et al., 2008]

Argumento de Posición de Saber/Autoridad		
	Esquema argumentativo	Cuestiones críticas
<i>Premisa Mayor</i>	La fuente <i>a</i> está en posición de conocer determinados hechos relacionados con un tema del dominio <i>S</i> que contiene la proposición <i>A</i> .	CC1: ¿Está <i>a</i> en posición de saber si <i>A</i> es verdad (falso)?
<i>Premisa Menor</i>	<i>a</i> afirma que <i>A</i> es verdad (falso).	CC2: ¿Es <i>a</i> una fuente fiable?
<i>Conclusión</i>	<i>A</i> es verdad (falso).	CC3: ¿De verdad <i>a</i> afirmó que <i>A</i> es verdad (falso)?
<b>Ejemplo</b>	Los profesionales médicos y científicos están en posición de conocer sobre la propagación de enfermedades infecciosas en espacios públicos. Los profesionales médicos y científicos afirman que usar mascarillas en espacios públicos puede reducir significativamente la propagación de enfermedades infecciosas. Por lo tanto, usar mascarillas en espacios públicos es una medida efectiva para prevenir la propagación de enfermedades infecciosas.	

**Tabla 2.2:** Esquema argumentativo de Autoridad con la información de sus premisas, conclusiones, conjunto de cuestiones críticas y una frase de ejemplo

Por otra parte, en el lenguaje natural hay diferentes tipos de diálogo según la intención del hablante [McBurney and Parsons, 2002]: diálogo persuasivo, de investigación, de negociación, informativo, deliberativo, etc. Además, existen los llamados cambios dialécticos, fenómenos que se dan cuando un argumento comienza siendo enmarcado en un tipo de diálogo, pero a medida que avanza la cadena de argumentación, va derivando en un tipo diferente de diálogo. Desde el punto de vista argumentativo, se debe evaluar la validez de estos cambios en el discurso, ya que indirectamente pueden llevar a manipulaciones e incluso generar falacias que lleven a conclusiones engañosas o parcialmente ciertas.

Argumento de Ejemplo		
Esquema argumentativo		Cuestiones críticas
<i>Premisa</i>	En este caso particular, el individuo $a$ tiene la propiedad $F$ y también la propiedad $G$ .	CC1: ¿Es verdadera la proposición afirmada en la premisa? CC2: ¿El ejemplo proporcionado apoya la generalización que se supone que ilustra?
<i>Conclusión</i>	Por lo tanto, generalmente, si $x$ tiene la propiedad $F$ , entonces también tiene la propiedad $G$ .	CC3: ¿Las circunstancias especiales del ejemplo afectan su generalización?
<b>Ejemplo</b>	En este caso particular, China ha implementado la censura en Internet y ha limitado el acceso a la información para sus ciudadanos, así como ha suprimido la libertad de expresión. Por lo tanto, en general, si un país implementa la censura en Internet, limitará el acceso a la información y suprimirá la libertad de expresión, yendo en contra de los principios de la democracia y violando los derechos humanos.	

**Tabla 2.3:** Esquema argumentativo de Ejemplo con la información de sus premisas, conclusiones, conjunto de cuestiones críticas y una frase de ejemplo

Argumento de Malgasto/Desperdicio		
Esquema argumentativo		Cuestiones críticas
<i>Premisa 1</i>	Si $a$ deja de intentar realizar $A$ ahora, todos los esfuerzos anteriores de $a$ para realizar $A$ habrán sido en vano.	CC1: ¿Es posible lograr $A$ ?
<i>Premisa 2</i>	Si todos los intentos anteriores de $a$ para realizar $A$ son en vano, eso sería algo malo.	CC2: Olvidando las pérdidas pasadas que no pueden recuperarse, ¿se debería realizar una reevaluación de los costos y beneficios de intentar realizar $A$ desde este momento en adelante?
<i>Conclusión</i>	Por lo tanto, $a$ debería continuar intentando realizar $A$ .	
<b>Ejemplo</b>	Si dejamos de intentar abordar el cambio climático ahora, todos los esfuerzos anteriores para mitigar sus efectos habrán sido en vano. Si todos nuestros esfuerzos anteriores para abordar el cambio climático resultan en vano, el planeta seguirá enfrentando consecuencias devastadoras como el aumento del nivel del mar, eventos climáticos extremos y la extinción de la vida silvestre. Por lo tanto, debemos seguir esforzándonos hacia un futuro sostenible y combatir activamente las causas y efectos del cambio climático.	

**Tabla 2.4:** Esquema argumentativo de Desperdicio con la información de sus premisas, conclusiones, conjunto de cuestiones críticas y una frase de ejemplo

## 2.2 La sociedad de la desinformación

En la actualidad, la sociedad de la información se enfrenta a un fenómeno preocupante y muy extendido: la propagación de información falsa o inexacta. Siguiendo la distinción llevada a cabo en [Stahl, 2006], dentro del ámbito de la desinformación encontramos dos conceptos con distintos matices (cuya diferencia se aprecia en inglés, ya que en castellano comparten término):

- La *misinformation* se refiere a la difusión involuntaria o no intencionada de información falsa o incorrecta. Puede ser el resultado de errores, malentendidos, sesgos, información obsoleta o imprecisa que se comparte sin intención de engañar.
- La *disinformation* es la difusión deliberada y planificada de información falsa, engañosa o manipulada con el propósito específico de influir, engañar o manipular a las personas. La desinformación es una herramienta estratégica utilizada, por ejemplo, en contextos políticos o de confrontación psicológica para sembrar confusión, desestabilizar o afectar la opinión pública.

Conseguir identificar con éxito la desinformación es una tarea costosa y a la vez complicada. Tanto a los verificadores de hechos humanos como a los verificadores automáticos, les cuesta detectar desinformación con un alto grado de certeza. Esto es debido a la complejidad de la tarea y a la falta de conjuntos de datos para entrenar sistemas que posean la "verdad última". No obstante, existen dos enfoques para contrarrestar la desinformación, tal y como se remarca en [Tay et al., 2022]:

- **Estrategia de *prebunking* o prevención anticipada:** implica aumentar de manera preventiva la conciencia de los ciudadanos sobre las técnicas de desinformación para hacerlos más resistentes a las noticias falsas.
- **Estrategia de *debunking* o refutación:** implica la corrección posterior de contenido engañoso que circula a través de los medios digitales.

El enfoque más eficaz actualmente es el *prebunking*, porque las tareas en el ámbito del procesamiento del lenguaje natural para detectar y corregir desinformación no son sencillas. En el área de la argumentación computacional, tareas como la extracción de los argumentos, sus límites, las relaciones entre ellos y las intenciones de cada uno entrañan gran dificultad. El principal problema es la existencia en el lenguaje natural de muchas ambigüedades, vaguedades, entimemas<sup>1</sup> y cambios dialécticos que hacen que la tarea no sea sencilla.

La detección de desinformación implica tareas de *fact-checking*. El objetivo principal del *fact-checking* es determinar si una afirmación es precisa y está respaldada por evidencia sólida o si, por el contrario, es engañosa, inexacta o falsa. Según [Musi et al., 2023], las tareas se dividen en dos tipos:

- ***Content-checking*:** verifica la precisión de la información y la exactitud de un contenido.
- ***Reason-checking*:** más enfocado a evaluar la validez y solidez de los argumentos utilizados en un texto.

Hoy en día, se busca realizar una transición desde el *content-checking* al *reason-checking*. Ya no solo es importante el contenido en sí, sino la lógica subyacente y el razonamiento que se usa para presentarlo.

Por otra parte, **la detección de postura o *Stance Detection*** juega un papel fundamental en la detección de noticias falsas al determinar el "punto de vista o postura" que un texto adopta hacia una afirmación. Esto es crucial para evaluar la veracidad de una afirmación, ya que la confiabilidad de una afirmación está estrechamente relacionada con el nivel de acuerdo expresado a favor o en contra en otros textos, especialmente aquellos provenientes de fuentes con alta credibilidad.

## 2.3 Procesamiento de lenguaje natural

---

La tarea de detectar desinformación es en su origen una tarea de procesamiento de lenguaje natural (PLN) ya que se parte de textos escritos o hablados en un determinado idioma. Antes de aplicar tareas más específicas de *Argument Mining*, se debe abordar un procesamiento del lenguaje natural como se menciona en [Montoro-Montarroso et al., 2023]. Las fases de esta tarea son:

<sup>1</sup>Argumentos incompletos o con partes implícitas

1. **El preprocesamiento del texto:** consiste en limpiar el texto de entrada, obteniendo la información más útil. Normalmente, se llevan a cabo los procesos de: tokenización, eliminación de *stopwords*, *stemming* y lematización. El proceso de tokenización es fundamental, ya que consiste en dividir el texto de entrada en unidades más pequeñas llamadas *tokens*. Los tokenizadores más destacados son: el Tokenizador de NLTK [Hardeniya et al., 2016], el de spaCy [Vasiliev, 2020], el WordPiece [Song et al., 2020], el SentencePiece [Kudo and Richardson, 2018] y el tokenizador de Hugging Face Transformers [Wolf et al., 2019] (los tres últimos usados típicamente en arquitecturas *Transformer* [Vaswani et al., 2017]).
2. **La extracción de características:** con el objetivo de identificar los rasgos básicos de los datos textuales que resulten importantes para la tarea. Se usan técnicas de etiquetado de partes del discurso según: sentimientos (dirigido a tareas de minería de opiniones), componentes o tipos de argumentos (centrándose en tareas de minería de argumentos), categorías léxicas o, incluso, se pueden llegar a generar bolsas de palabras (*bag-of-words*) [Qader et al., 2019] que tienen en cuenta la frecuencia de aparición de las unidades lingüísticas.
3. **La representación de textos numéricamente:** consiste en representar los *tokens* previamente obtenidos según vectores numéricos, capturando el significado y la relación semántica entre las palabras en función de su contexto global. Una de las técnicas más usadas actualmente son las incrustaciones de palabras (*word embeddings*) con populares algoritmos de aprendizaje profundo como el Word2Vec [Church, 2017], Skip-Gram [Guthrie et al., 2006], Continuos Bag of Words [Wang et al., 2017] y el algoritmo GloVe [Pennington et al., 2014]. Este último está más dedicado a conjuntos de datos pequeños por su mayor aprovechamiento de información estadística.

Sin embargo, el principal reto de llevar a cabo tareas de procesamiento de lenguaje natural es conservar el orden de las palabras en su contexto original. La pérdida de gran parte de esta información puede provocar un rendimiento insuficiente del modelo desarrollado. Por tanto, nos interesa saber el contexto global de cada palabra y respetar su orden, ya que esta cuestión es clave en el sentido y significado global. La obtención de esa dependencia a largo plazo es difícil de conseguir, pero existen algunas arquitecturas que demuestran ser capaces. Estas arquitecturas son los llamados modelos secuenciales.

Los modelos secuenciales trabajan en el procesamiento de secuencias al procesar datos secuenciales en orden, es decir, un dato (palabra) cada vez. Destacan las siguientes arquitecturas de modelos secuenciales:

- **Las Redes Neuronales Recurrentes (RNN)** [Yang et al., 2020]: tratan el texto como una secuencia capturando la dependencia a largo plazo en secuencias de palabras, gracias a sus conexiones recurrentes que les permiten mantener un estado interno que representa la historia de las entradas previas. No obstante, en frases muy largas, surgen los problemas del gradiente explosivo y el desvanecimiento del gradiente que impiden que el modelo llegue a una solución óptima. A pesar de esto, existen dos variantes de las RNN que evitan parcialmente estos problemas: las LSTM (Long Short Term Memory) y las GRU (Gated Recurrent Unit) y, además, demuestran un mayor aprendizaje a largo plazo.

En la Figura 2.2<sup>2</sup> observamos una comparativa de la estructura y los componentes de estos tipos de RNN. Los componentes básicos de las RNN son: el dato actual de

<sup>2</sup>Fuente: <https://medium.com/@sachinsoni600517/unlocking-the-power-of-gated-recurrent-unit-gru-understanding-the-innovative-architecture-e2d80d0e91b6>

la secuencia ( $x_t$ ), el estado oculto del paso anterior ( $h_{t-1}$ ) que actúa como una memoria de la red y una función de activación hiperbólica tangente ( $\tanh$ ) que calcula el nuevo estado oculto ( $h_t$ ). Así, podemos ver cómo en la figura, la estructura de la Simple RNN solo dispone del módulo con la función tangente, mientras que en las redes LSTM y GRU están compuestas por varios módulos.

Las LSTM (Figura 2.2) cuentan con un nuevo componente: el estado de la celda actual ( $c_t$ ) y anterior ( $c_{t-1}$ ). La modulación con la función tangente y sigma, además del control de flujo de información con las diferentes puertas o gates (puerta de olvido ( $f_t$ ), puerta de entrada ( $i_t$ ) y puerta de salida ( $o_t$ )) permite mantener la memoria a largo plazo a través de la secuencia.

Por su parte las GRU (Figura 2.2), aunque funcionan de manera similar a las LSTM, tienen menos componentes y son más eficientes. Disponen de dos puertas: la puerta de reinicio ( $r_t$ ) y la puerta de actualización ( $z_t$ ), cuyo funcionamiento también permite manejar dependencias a largo plazo.

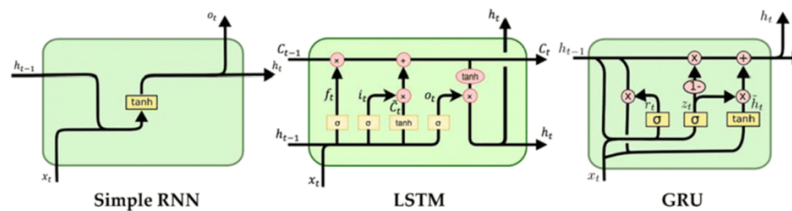


Figura 2.2: Comparativa de Redes Neuronales Recurrentes.

- **Las Redes Neuronales Convolucionales 1D (CNN 1D** [Yin et al., 2017], Figura 2.3)<sup>3</sup>: aunque estas arquitecturas se popularizaron inicialmente en el campo de la visión por computador para procesar imágenes, también se pueden utilizar para procesar secuencias unidimensionales, como texto o series temporales, mediante el uso de operaciones de convolución en una dimensión (1D).

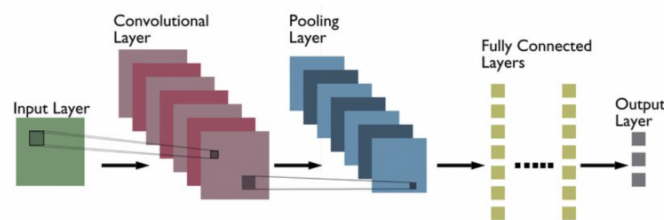


Figura 2.3: Arquitectura de las Redes Neuronales Convolucionales.

- **Los Transformers** [Gillioz et al., 2020] (Figura 2.4)<sup>4</sup>: estos modelos también basados en redes neuronales han mostrado un excelente rendimiento en muchas tareas de PLN (incluyendo la minería de argumentos). Son capaces de capturar el contexto semántico y sintáctico de las palabras de manera más efectiva que los modelos previos secuenciales y más tradicionales. Hay dos familias de transformers: los basados en codificadores, más conocidos por funcionar como clasificadores (modelos como BERT, DistiBERT, RoBERTa..) [Adoma et al., 2020]; y los basados en decodificadores, denominados generadores (como GPT (*Generative Pre-trained Transformer* [Yenduri et al., 2024])).

<sup>3</sup>Fuente: <https://medium.com/@navarai/unveiling-the-diversity-a-comprehensive-guide-to-types-of-cnn-architectures-9d70da0b4521>

<sup>4</sup>Fuente: <https://www.aprendemachinelearning.com/como-funcionan-los-transformers-espanol-nlp-gpt-bert/>

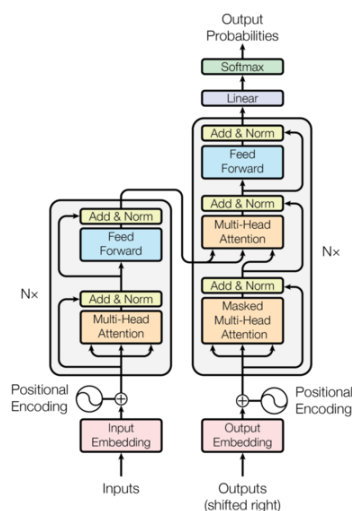


Figura 2.4: Arquitectura Transformer.

Tras la publicación de [Vaswani et al., 2017] los modelos Transformers se dieron a conocer al mundo volviéndose extremadamente populares. Más recientemente se han aplicado con gran éxito en el ámbito del Procesamiento del Lenguaje Natural debido a las grandes ventajas que proporcionan. A diferencia de las RNN, los modelos Transformers gracias a sus mecanismos de atención, son capaces de capturar relaciones de largo alcance sin ninguna limitación en la longitud y complejidad de dependencias entre palabras. Además, permiten la paralelización eficiente ya que pueden procesar secuencias de entrada de manera paralela, siendo más funcionales que los modelos anteriores.

## 2.4 Argument Mining

Tras analizar el estado del arte en el PLN, nos centramos en el campo de la argumentación computacional. La minería de argumentos o *Argument Mining*, es una subárea de la argumentación computacional que se enfoca específicamente en la extracción de argumentos y estructuras argumentativas de texto no estructurado.

El objetivo es identificar automáticamente los argumentos relevantes de un texto y detectar sus propiedades causales de premisas y conclusiones, sus relaciones de ataque y soporte, e incluso detectar los patrones complejos del razonamiento humano basándose en los esquemas argumentativos mencionados anteriormente.

Concretamente, para la tarea de detección de posturas (previamente explicada en la sección 2.2) observamos como en [Kotonya and Toni, 2019] se utiliza un método relevante para detectar noticias falsas. Se basa en el uso de *Bipolar Argumentation Frameworks* (BAFs<sup>5</sup>) para representar las relaciones argumentativas entre diferentes textos y calcular las fuerzas dialécticas de los argumentos. La fuerza de cada argumento representa el efecto que sus argumentos atacantes y defensores provocan en él.

Se podría asumir que las afirmaciones falsas se debilitarán por la fuerza y cantidad de los argumentos que las atacan, lo que supondría una baja fuerza dialéctica según este algoritmo. Esto se debe a que las afirmaciones falsas generalmente carecen de argumentos de apoyo y tienen muchos atacantes en su contra. Sin embargo, esta suposición no siempre es válida, especialmente debido a la existencia de las llamadas cámaras de eco

<sup>5</sup>Frameworks de Argumentación Bipolares donde cada argumento se representa como un nodo en un grafo y las relaciones entre los argumentos se representan mediante arcos dirigidos que indican ataques o defensas entre ellos

(también conocidas como "silos"). Este es un fenómeno ampliamente extendido en las redes sociales en el que se hace evidente cómo los argumentos falsos pueden estar respaldados por otros argumentos igualmente engañosos. Como consecuencia, una afirmación falsa podría llegar a tener una alta fuerza dialéctica según el algoritmo mencionado ya que está respaldada por varios argumentos engañosos en esa cámara de eco.

En el marco de este trabajo, se ha llevado a cabo una extensa investigación sobre qué modelos y arquitecturas se suelen usar en tareas relacionadas con la argumentación computacional en textos. Entre los proyectos investigados se encuentran tanto modelos más tradicionales [Iranzo-Sánchez and Ruiz-Dolz, 2019], como modelos últimamente más populares basados en Transformers [Ruiz-Dolz et al., 2021] e incluso experimentos en los que se han combinado las dos generaciones [Goffredo et al., 2023]. La Tabla 2.5 muestra una comparativa y recopilación de todas estas arquitecturas que a continuación se explican de forma más detallada.

Modelos tradicionales		Modelos basados en Transformers
<ul style="list-style-type: none"> <li>• Parsing Algorithms</li> <li>• Textual Entailment Suites</li> <li>• Logistic Regression</li> <li>• Naive Bayes</li> <li>• Gradient Tree Boosting</li> <li>• Support Vector Machines (SVMs)</li> <li>• Redes Neuronales Recurrentes (RNN)</li> </ul>		<ul style="list-style-type: none"> <li>• BERT</li> <li>• XLNET</li> <li>• RoBERTa</li> <li>• DistilBERT</li> <li>• ALBERT</li> <li>• Zeroshot prompting GPT-3.5-TURBO and GPT-4</li> </ul>
<b>Mejores arquitecturas</b>	<b>SVMs</b> RNN (concretamente LSTMs y BiLSTMs)	<b>RoBERTa</b>

Tabla 2.5: Comparativa de arquitecturas y modelos destacados

### 2.4.1. Modelos de lenguaje tradicionales

En [Iranzo-Sánchez and Ruiz-Dolz, 2019] se experimenta con modelos de lenguaje más tradicionales para llevar a cabo tareas de detección de ironías en mensajes textuales. Destacan los siguientes modelos:

- ***Parsing Algorithms*** (Algoritmos de Análisis Sintáctico): utilizan parsers sintácticos y semánticos para analizar la estructura de las oraciones.
- ***Textual Entailment Suites*** (Conjuntos de Inferencia Textual): diseñados para evaluar la relación de inferencia entre pares de oraciones.
- ***Logistic Regression y Naive Bayes***: modelos tradicionales de clasificación que pueden ser efectivos en ciertos contextos, pero pueden ser superados por modelos más complejos en tareas de PLN.
- ***Gradient Tree Boosting***: utilizado en tareas de clasificación y regresión, puede ser efectivo pero puede tener dificultades con datos de alta dimensionalidad y texto no estructurado.
- ***Support Vector Machines (SVMs)***: tienen capacidad para manejar eficientemente conjuntos de datos de alta dimensionalidad y generalizan bien a partir de datos de entrenamiento limitados.
- ***Redes Neuronales Recurrentes (RNN)***: modelos bastante efectivos como hemos comentado previamente, ya que son capaces de capturar dependencias a largo plazo en secuencias de palabras.



De los modelos tradicionales comentados, aquellos que obtienen mejores resultados son las *Support Vector Machines* (SVMs) y las Redes Neuronales Recurrentes. Concretamente entre las RNN, destacan las siguientes arquitecturas especializadas en el procesamiento de secuencias de datos:

- **Long Short-Term Memory (LSTMs):** tipo de red neuronal recurrente que supera el problema de la "desaparición del gradiente", típico de las RNN tradicionales. Están compuestas por celdas de memoria que pueden retener información durante largos períodos de tiempo y tienen unas puertas especiales (de entrada, de salida y de olvido) que permiten que la red aprenda qué información recordar y qué olvidar durante el procesamiento de una secuencia.
- **Bidirectional Long Short-Term Memory (BiLSTM):** son una extensión de las LSTMs que procesan la secuencia de entrada en dos direcciones: hacia adelante y hacia atrás. Al procesar la secuencia de entrada en ambas direcciones, las BiLSTMs capturan tanto el contexto pasado como el futuro de cada palabra en el texto.

La principal ventaja de usar modelos más tradicionales es que no se requiere gran cantidad de datos de entrada, ya que generalizan bastante bien. Además, su entrenamiento no es muy costoso en comparación con modelos más potentes y actuales. Sin embargo, es evidente que tienen dificultad al capturar relaciones semánticas complejas y poseen limitaciones en la captura de dependencias a largo plazo, demostrando que tienen un rendimiento mejorable en tareas complejas de PLN.

#### 2.4.2. Grandes modelos de lenguaje basados en Transformers

A partir de 2018, hubo una gran revolución en el ámbito del Procesamiento de Lenguaje Natural ya que se comenzaron a utilizar ampliamente los Grandes Modelos de Lenguaje (LLM) basados en Transformers. Uno de los principales factores que contribuyeron a la popularidad de estos LLM fue su capacidad para el "*fine-tuning*", es decir, adaptar modelos previamente entrenados a tareas específicas ajustando los pesos de la red neuronal en función de los datos de entrenamiento disponibles. En [Ruiz-Dolz et al., 2021], podemos ver cómo esta técnica se aplica a gran variedad de modelos como:

- **BERT (Bidirectional Encoder Representations from Transformers):** desarrollado por Google.
- **XLNET:** también desarrollado por Google, capaz de modelar relaciones tanto hacia adelante como hacia atrás en una secuencia.
- **RoBERTa:** variante de BERT más refinada desarrollada por Facebook que utiliza una arquitectura de transformers pre-entrenada y se ajusta con un enfoque más agresivo de pre-entrenamiento y datos más grandes. RoBERTa mejora el rendimiento en una variedad de tareas de PLN, incluido el *Argument Mining*, al capturar mejor el contexto y la semántica del texto.
- **DistilBERT y ALBERT:** DistilBERT es una versión más ligera y comprimida de BERT desarrollada por Hugging Face, mientras que ALBERT reduce significativamente el número de parámetros y mejora la eficiencia computacional sin sacrificar el rendimiento.
- **Zeroshot prompting GPT-3.5-TURBO y GPT-4:** modelos de lenguaje de generación de texto desarrollados por OpenAI. Aunque no fueron específicamente diseñados para tareas de *Argument Mining*, su capacidad para generar texto coherente y relevante basado en un contexto de entrada puede ser ampliamente aprovechada.

Estos modelos basados en Transformers son más potentes y demuestran, en general, mejores resultados capturando un rango de información más amplio gracias a su arquitectura basada en mecanismos de atención. No obstante, su entrenamiento es muy costoso si no se aplican técnicas como el *fine-tuning* y, además, es necesario disponer de un gran conjunto de datos.

En definitiva, dada esta comparativa, podríamos concluir que uno de los pilares del estado del arte de la argumentación computacional para detectar desinformación son los LLMs basados en Transformers. Sin embargo, centrándonos en la tarea de detectar desinformación o problemas de razonamiento en textos, se constata que existen limitaciones derivadas de usar únicamente LLMs. Este aspecto se debate en la sección 2.5.

### 2.4.3. Chatbots basados en grandes modelos de lenguaje

Los avances logrados con los grandes modelos de lenguaje (LLMs) han impulsado el desarrollo de gran cantidad de chatbots. Estas potentes herramientas se han convertido en el día a día de muchas personas y han resultado ser de gran utilidad por su gran conocimiento y personalización de contenido mostrado a través de una interfaz conversacional intuitiva y accesible. Por tanto, no es descabellado plantear el uso de estas herramientas para potenciar un modelo de Inteligencia Artificial que se ajuste con un buen rendimiento a una tarea específica.

Gracias a estas tecnologías, encontramos nuevas maneras de suministrar información y ajustar el conocimiento de los LLMs. Como podemos apreciar en [Minaee et al., 2024], a la ya conocida técnica de *fine-tuning* se le unen otras técnicas de recuperación y de ajuste aplicadas a los LLMs:

- **"Priming" o "Prompting"**: a través de un *"prompt"* (texto inicial), se le ofrece al modelo una introducción o ejemplos relevantes que sirven como contexto. De esta manera, el modelo se vuelve más preciso y está más enfocado a la tarea.
- **"Fine-tuning"**: re-entrenamos un modelo pre-entrenado ajustándolo a un conjunto de datos específicos o a la tarea que queremos realizar. Esta técnica es de las más costosas, porque implica ajustar los pesos del modelo para que se adapten mejor a la tarea deseada.
- **"Retrieval-Augmented Generation" (RAG)**: es una arquitectura compuesta por técnicas de recuperación y generación de texto. Utiliza un modelo de recuperación para encontrar documentos relevantes a partir de una consulta y luego un modelo de generación para generar texto basado en la información recuperada. Al contrario que en *fine-tuning*, no se lleva a cabo un re-entrenamiento de la red, pero sí que se logra generar texto más específico sobre la tarea objetivo. Esta es la técnica que usan muchos de los modelos GPT conectados a internet.

En [Gorur et al., 2024] se intentan clasificar las relaciones de ataque y soporte entre argumentos a través de un suministro de *prompts*. Las pruebas realizadas con esta técnica aplicada a modelos como LLAMA [Touvron et al., 2023] y Mistral [Jiang et al., 2023], demuestra mejores resultados que los vistos en [Ruiz-Dolz et al., 2021] donde se usan técnicas de *fine-tuning* en modelos de la familia BERT, comentados anteriormente. En la (Figura 2.5)<sup>6</sup>, podemos ver un ejemplo de cómo se ha usado la técnica basada en prompts para determinar el tipo de relación entre argumentos. Concretamente, al modelo se le suministra un *"Primer"* que contiene ejemplos resueltos de la tarea que tiene que

<sup>6</sup>Fuente: Artículo "Can Large Language Models perform Relation-based Argument Mining?" [Gorur et al., 2024]

realizar y, seguidamente, se le suministra un "Prompt" con un ejemplo incompleto. De este modo, el modelo es capaz de clasificar el tipo de relación entre los argumentos del "Prompt" basándose en los ejemplos previos del "Primer".

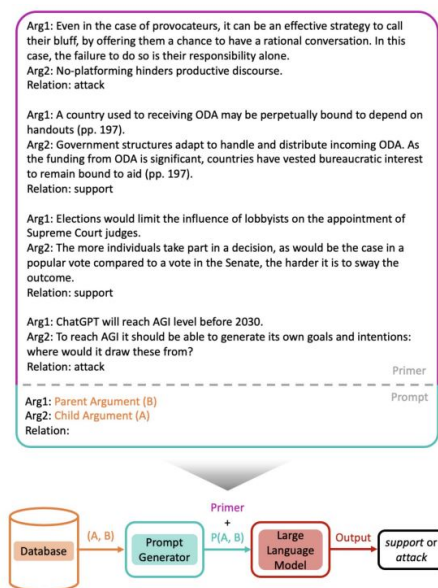


Figura 2.5: Técnica de priming y prompting aplicada a un Chatbot basado en LLM.

Por tanto, podemos observar como las tecnologías basadas en chatbots tienen comportamientos inteligentes a la espera de ser aprovechados. Otra muestra de ello es la clasificación de argumentos en esquemas argumentativos llevada a cabo en [de Sousa et al., 2024], donde se implementa un clasificador a través de un framework de código abierto para construir chatbots y asistentes conversacionales, llamado Rasa <sup>7</sup>.

## 2.5 Crítica al estado del arte

Aunque el estado del arte anteriormente presentado es bastante amplio y prometedor, en ciertos casos, se revelan limitaciones significativas que afectan la eficacia y la fiabilidad de tales enfoques. Estas limitaciones se exponen a continuación.

### 2.5.1. Limitaciones de los LLMs

Tras la investigación llevada a cabo en el campo de la argumentación computacional y los LLMs, se ha descubierto que los grandes modelos de lenguaje poseen ciertas limitaciones. En muchos de los artículos científicos vistos [Goffredo et al., 2023], se implementan clasificadores de argumentos mediante LLMs que se basan en la estructura o patrones de las palabras que componen la entrada al modelo, pero nunca se profundiza en qué razonamiento se ha seguido para saber si el argumento es válido (o si es una falacia y de qué tipo). Es decir, no se presta atención a la lógica subyacente que hace que el argumento sea veraz, sino simplemente se detecta y clasifica según su estructura.

Este problema se puede ver fácilmente reflejado en las dificultades que tienen los LLMs para generar una secuencia de argumentos siguiendo un razonamiento adecuado cuando intentan respaldar un tema. Como se refleja en [El Baff et al., 2019], muchos de

<sup>7</sup><https://rasa.com/>

los LLMs son muy buenos generando grandes cantidades de texto que proporcionan información, pero cuando tienen que razonar de forma extensa, demuestran no seguir con un razonamiento lógico hasta el final e incluso se llegan a contradecir. Este problema se evidencia en la mayoría de LLMs actuales, aunque gradualmente se están desarrollando modelos de lenguaje más avanzados, como es el caso de GPT4, que aumentan su razonamiento lógico. Sin embargo, aún queda mucho margen de mejora, ya que modelos como GPT4 siguen presentando alucinaciones, lo que significa que ocasionalmente generan información incorrecta o inventada que, aunque a menudo parece plausible, no es precisa.

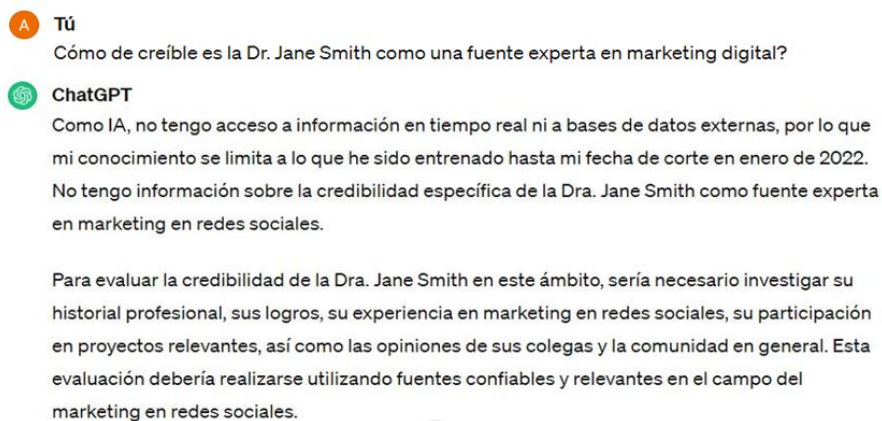
No obstante, la argumentación computacional es clave en este contexto ya que puede servir como apoyo a los LLMs para detectar y generar una línea de razonamiento más adecuada y veraz. Así que, un enfoque más completo sería integrar técnicas típicas de argumentación computacional en los grandes modelos de lenguaje estado del arte del campo de PLN. Esta combinación concedería a los LLMs la capacidad de aprender de características argumentativas potenciando su rendimiento.

Concretamente, en [Ruiz-Dolz and Lawrence, 2023] se presenta un sistema de detección de falacias innovador basado en el uso de LLMs y esquemas argumentativos. Se usan dos datasets: uno con falacias clasificadas según el tipo y otro dataset de validación basado en esquemas argumentativos. El uso del segundo dataset resulta un enfoque novedoso, ya que otorga directamente al sistema información sobre características argumentativas de las cuales puede aprender. Cada esquema argumentativo se relaciona con un tipo de falacia. Por cada esquema argumentativo, se guardan dos ejemplos de frases: una que no es falacia (ya que contesta todas las cuestiones críticas del esquema adecuadamente) y otra que sí está clasificada como falacia (debido a que no supera alguna cuestión crítica). De este modo, en el artículo, se desarrollan modelos capaces de identificar falacias de forma precisa y se concluye que no se puede confiar exclusivamente en los LLMs para abordar estas tareas, destacando la necesidad de respaldar su uso con la argumentación computacional.

Por otra parte, otra limitación que poseen los LLMs son sus elevados costos computacionales durante el entrenamiento y el refinamiento. El entrenamiento desde cero de un LLM requiere meses en grandes clústeres de GPU, demandando numerosos recursos económicos y causando un gran impacto en la huella de carbono. Las técnicas de refinamiento comentadas anteriormente son una mejoría de esta limitación, pero el coste computacional en alguna de ellas sigue siendo elevado como es el caso del *fine-tuning*, una de las técnicas más utilizadas y que ha demostrado obtener buenos resultados. Por tanto, es necesario investigar otras técnicas de refinamiento en LLMs que sean capaces de reducir estos costos computacionales sin perjudicar al rendimiento del modelo. Este es el caso del *prompt engineering*, técnica que según [Gorur et al., 2024], ha demostrado alcanzar resultados comparables o incluso superiores al *fine-tuning* en tareas de minería de argumentos.

### 2.5.2. Sistema Multiproceso con conexión a Fuentes Externas

Aunque los LLM son grandes generadores de texto, a menudo presentan deficiencias para recuperar información específica ya que este conocimiento puede no estar en su dominio de entrenamiento. Esto puede resultar en respuestas inexactas o irrelevantes en ciertos contextos. Por ejemplo, en la Figura 2.6 se hace patente la limitación de ChatGPT 3.5 al no tener acceso directo a internet ni a bases de datos en tiempo real. Su conocimiento se basa en datos de entrenamiento de información disponible hasta enero de 2022.



**Figura 2.6:** Ejemplo de limitación de ChatGPT 3.5 al intentar responder a un prompt sin éxito ya que el LLM no tiene acceso a fuentes con información de ese dominio

Por otra parte, en numerosos artículos se detectan ciertos problemas o asunciones que pueden llevar a la implementación de un modelo de detección de desinformación que no es completamente fiel a la realidad.

En primer lugar, al trabajar solo con una fuente de información, el resultado de cualquier modelo puede llegar a ser autoritario y estar sesgado por los propios sesgos inherentes a los datos con los que se ha entrenado el modelo. En segundo lugar, a menudo los usuarios dan por sentado que los resultados que se consiguen con los LLMs son completamente creíbles, sin un razonamiento crítico sobre el contenido generado ni poder contrastar las fuentes. En tercer y último lugar, muchas veces no se tiene en cuenta la complejidad del lenguaje natural. Al querer dictaminar la veracidad de un argumento nos planteamos clasificarlo binariamente (verdadero o falso). Sin embargo, una clasificación multiclase en intervalos de veracidad o un valor numérico representativo sería más fiel a la realidad.

Es por ello que, en artículos como [de Sousa et al., 2024], se proponen sistemas multiagentes o multitareas para abordar tareas de PLN. Por tanto, se podrían potenciar los LLM conectándolos a diferentes fuentes de información en cada proceso. Así el modelo podría acceder a diferentes fuentes de datos durante su funcionamiento, lo que aumentaría su capacidad para recuperar información específica y responder de manera más precisa a las consultas.

## 2.6 Propuesta

---

Una vez analizado el estado del arte, se observa como los LLMs resultan modelos potentes e idóneos para abordar tareas del ámbito de la argumentación computacional, como es la detección de desinformación.

No obstante, poseen ciertas limitaciones como: la carencia de seguir un razonamiento lógico de principio a fin, la ausencia de técnicas argumentativas aplicadas a su funcionamiento y la incapacidad de recuperar información específica de fuentes y bases de datos externas. Para conseguir un sistema de detección de desinformación con unos resultados correctos y eficientes, se propone mejorar y potenciar el uso de LLMs solucionando estas barreras.

Para ello, la línea de trabajo irá dirigida, por una parte, a dotar a los LLMs de una nueva dimensión lógica basada en conceptos de la teoría de la argumentación (como pueden

ser los esquemas argumentativos). Por otra parte, se tendrán en cuenta diversas fuentes de información con la intención de descubrir la veracidad y contrastar la finalidad tanto lógica como factual de los argumentos. De esta manera, el sistema de detección de desinformación clasificará el argumento de entrada en un esquema argumentativo concreto y proporcionará su correspondiente patrón de razonamiento al LLM para que este sea capaz de evaluar el nivel de veracidad del argumento a través de una justificación razonada y respaldada con datos externos de diferentes fuentes de información, utilizando para ello las cuestiones críticas asociadas a cada esquema argumentativo.

Además, se propone mejorar el rendimiento y eficiencia de este sistema, desarrollando un sistema multiproceso donde la recuperación de información de cada fuente se almacene en un proceso distinto, de forma que se pueda realizar en paralelo. De este modo, se sigue una estrategia que permite el desarrollo modular, la distribución del conocimiento, la flexibilidad, la adaptabilidad y la escalabilidad del sistema en un futuro.

---

## CAPÍTULO 3

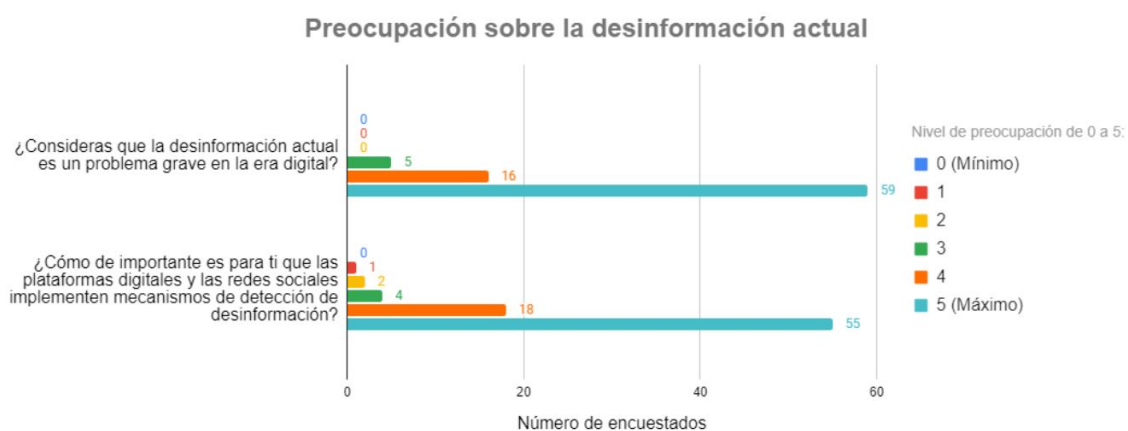
# Análisis del Problema

---

En la actualidad, combatir el problema de la difusión de desinformación se ha convertido en un desafío importante para la sociedad. La proliferación de información en el entorno digital ha creado un contexto en el que los ciudadanos se enfrentan a un flujo constante de datos, muchas veces contradictorios, que dificultan la capacidad de discernimiento crítico. Este problema no solo afecta a la percepción de la realidad de las personas, sino que también tiene repercusiones en áreas cruciales como la salud pública, el medio ambiente, la política y otros ámbitos relacionados con el bienestar de la sociedad.

En el marco inicial de este proyecto, se ha realizado una encuesta exploratoria con el objetivo de comprender y detectar las necesidades y percepciones de la población, reconociendo y contextualizando la magnitud y el alcance del problema de la propagación de desinformación.

La encuesta se ha llevado a cabo con una muestra representativa de 80 personas adultas con edades comprendidas entre los 18 y los 65 años. Tras analizar las respuestas, se ha identificado un alto nivel de preocupación por la propagación de desinformación en la época actual. Los encuestados no solo han considerado crítico el problema, sino que también han destacado la importancia de contar con mecanismos de detección de desinformación en plataformas digitales y redes sociales. En la Figura 3.1, se observa como en una escala de 0 (Nada) a 5 (Mucho), la mayoría de encuestados sitúan sus respuestas en el nivel 5, resaltando el problema que supone la desinformación y la necesidad de herramientas que la controlen.



**Figura 3.1:** Valoración del nivel de preocupación sobre la propagación de desinformación

Los encuestados indicaron en qué ámbitos creían que la desinformación podría afectarles de manera significativa. Según la Figura 3.2, los ámbitos que se consideran más afectados por la desinformación son: la educación y el conocimiento de las personas, la

salud pública y el bienestar social, la confianza pública y el mantenimiento de instituciones públicas, la acción climática y la percepción del cambio climático, la igualdad de género y la reducción de desigualdades. Por tanto, es evidente la preocupación sobre los ámbitos anteriores y el efecto que la desinformación provoca sobre ellos, obstaculizando el progreso y la consecución de diversos Objetivos de Desarrollo Sostenible (ODS). En el Anexo A, se adjunta un esquema con la relación de los ODS vinculados con este proyecto.



**Figura 3.2:** Valoración de los ámbitos más afectados por la desinformación

En conclusión, la información recogida y analizada a partir de la encuesta exploratoria demuestra la necesidad de lograr soluciones efectivas para combatir la desinformación y promover una sociedad mejor informada.

### 3.1 Especificación de requisitos

En este proyecto se realiza el diseño e implementación de un sistema detector de desinformación con los siguientes requisitos funcionales:

Requisito	Descripción	Justificación
<b>Clasificación de Argumentos</b>	Diseñar un módulo de clasificación de argumentos que identifique el esquema argumentativo o patrón de razonamiento seguido en dicho argumento.	Este módulo proporcionará al sistema unas pautas para llevar a cabo una evaluación estructurada que estandarice y mejore la precisión y fiabilidad del sistema de detección de desinformación.
<b>Evaluación de Argumentos</b>	Diseñar un módulo que evalúe la veracidad del argumento de entrada mediante preguntas críticas y justificaciones respaldadas por contextualización externa.	Este módulo permitirá al sistema una evaluación estructurada y basada en evidencias que mejore la precisión y fiabilidad del sistema de detección de desinformación.



<b>Mejora del LLM con Dimensión Lógica</b>	Incorporar conceptos de la teoría de la argumentación basada en el uso de esquemas argumentativos, para dotar a los LLMs de una nueva dimensión lógica que permita un análisis más detallado, riguroso y sistemático de los argumentos.	Esto permitirá minimizar las limitaciones de los LLMs aumentando la capacidad de razonamiento lógico y mejorando la evaluación de la veracidad de los argumentos.
<b>Integración de Fuentes de Información Diversas</b>	Implementar un subsistema de recuperación de información que considere diversas fuentes externas, permitiendo la verificación y contraste de los datos en el funcionamiento del LLM.	La incorporación de múltiples fuentes permitirá que la evaluación de la veracidad sea más precisa, variada y esté respaldada por datos específicos que pueden no ser conocidos por el LLM utilizado como base.
<b>Sistema multi-proceso para la Recuperación de Información</b>	Desarrollar un sistema multitarea donde la recuperación de información de cada fuente se realice en paralelo, mejorando la eficiencia y el rendimiento.	Un sistema basado en programación concurrente permitirá una mayor rapidez en la recopilación y procesamiento de datos, facilitando un análisis más ágil y escalable.
<b>Facilidad de Uso a través del desarrollo de una Herramienta Web</b>	Implementar una interfaz web que permita a los usuarios ingresar el argumento a evaluar, analizar su veracidad y recibir un valor cuantitativo de veracidad junto con una justificación cualitativa.	La herramienta web facilitará el acceso a la funcionalidad del sistema, proporcionando una mejor experiencia de usuario.

**Tabla 3.1:** Requisitos funcionales y técnicos identificados para el desarrollo del sistema de detección de desinformación, detallando las descripciones y justificaciones correspondientes.



---

## CAPÍTULO 4

# Diseño de la solución

---

En este capítulo se analiza el diseño general del sistema propuesto para la detección de desinformación. La explicación se desarrolla siguiendo dos niveles de abstracción: inicialmente, se ofrece una visión general del sistema, y posteriormente, se detallan los aspectos específicos y más técnicos del diseño planteado.

### 4.1 Arquitectura del sistema

---

El propósito de este proyecto es desarrollar un sistema para la detección de desinformación, fundamentado en el uso de LLMs y técnicas de computación argumental. El sistema desarrollado es capaz de analizar la entrada de un argumento escrito en inglés y seguir un proceso de razonamiento identificando los posibles factores en los que presenta deficiencias, con el objetivo final de estimar como salida un valor cuantitativo que indique su nivel de veracidad.

En la Figura 4.1, se presenta un esquema de la arquitectura del sistema. Se aprecia como el sistema está formado por 2 módulos o subsistemas principales:

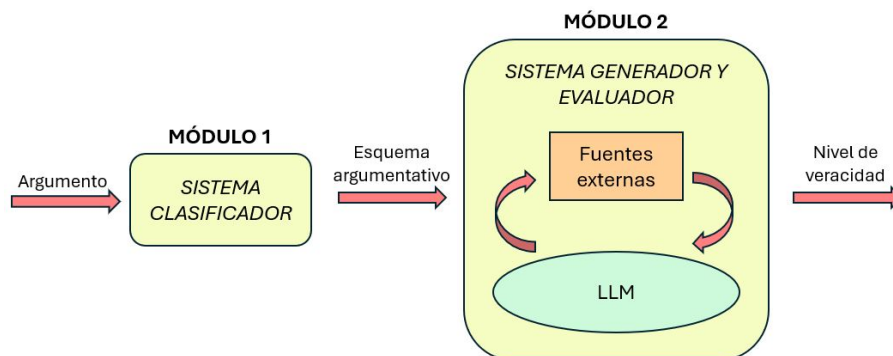


Figura 4.1: Esquema del diseño de la arquitectura del sistema

- **Módulo 1: "Sistema clasificador en esquemas argumentativos"**. Este módulo recibe como entrada una frase (escrita en inglés) que representa un argumento y clasifica dicho argumento en un esquema argumentativo de la teoría de la argumentación.
- **Módulo 2: "Sistema generador y evaluador del nivel de veracidad"**. Este módulo toma como entrada la salida del módulo anterior. La clase predicha por el Módulo 1 corresponde con un esquema argumentativo que contiene información sobre el tipo de razonamiento lógico utilizado en el argumento de entrada. Conociendo el

tipo de razonamiento que sigue la frase, se establecen una serie de cuestiones críticas que evalúan la veracidad del argumento. Finalmente, mediante el uso de un Modelo de Lenguaje Grande (LLM) con contextualización externa, se proporciona como salida una justificación cualitativa y un valor cuantitativo que indica su nivel de veracidad.

## 4.2 Diseño detallado

A continuación, se profundiza paso a paso en el diseño de cada módulo del sistema, representado de forma más detallada y en su totalidad en la Figura 4.2

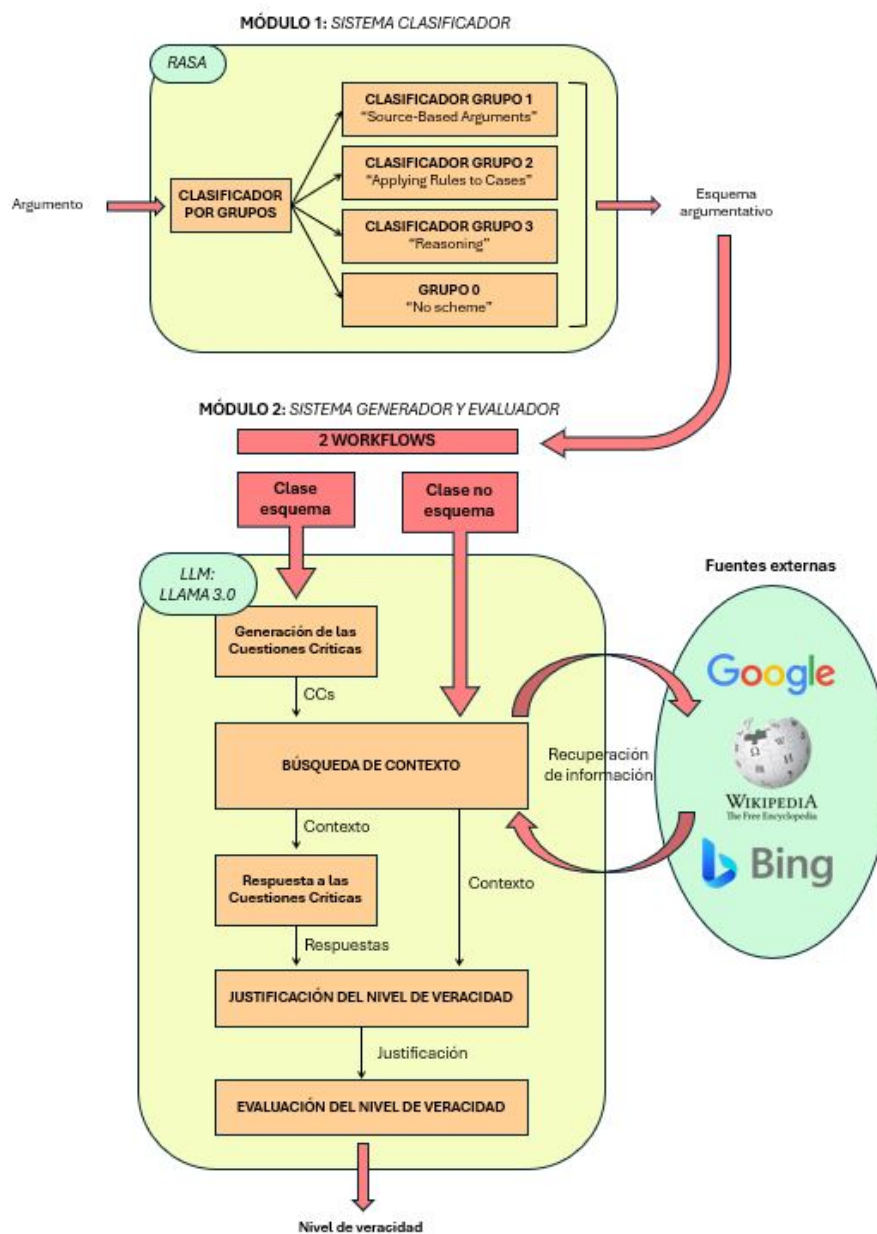


Figura 4.2: Esquema detallado del diseño de la arquitectura del sistema

### 4.2.1. Módulo 1: Sistema clasificador en esquemas argumentativos

Este módulo proporciona al sistema las directrices para ser capaz de detectar desinformación siguiendo un razonamiento basado en la teoría de la argumentación. Para ello, se implementa un sistema clasificador que, dado un argumento escrito en inglés, lo clasifica según el esquema argumentativo al que pertenece según la teoría de la argumentación.

La estrategia utilizada consiste en el desarrollo de un clasificador de dos capas, ya que inicialmente se trabajan con 19 clases diferentes de esquemas argumentativos que, aplicando aspectos teóricos de argumentación, se pueden agrupar en 4 grandes familias o grupos argumentativos. Se han utilizado los esquemas argumentativos más comunes del lenguaje natural encontrados en el libro de la teoría de la argumentación de Walton [Walton et al., 2008] e incluidos en el dataset "NLAS-MULTI corpus" [Ruiz-Dolz et al., 2024]. Más adelante, en los capítulos 5 y 6, se explicará en detalle el dataset utilizado, el entrenamiento llevado a cabo con la tecnología de RASA y el rendimiento más óptimo de este clasificador de doble capa en comparación con una versión experimental de una sola capa.

Por lo tanto, el sistema clasificador general está compuesto por un total de 4 clasificadores internos (Figura 4.3). En la primera capa, se encuentra el clasificador en grupos argumentativos y, en la segunda capa, los clasificadores en esquemas argumentativos de cada grupo. El clasificador de grupos puede clasificar en 4 categorías o familias argumentativas:

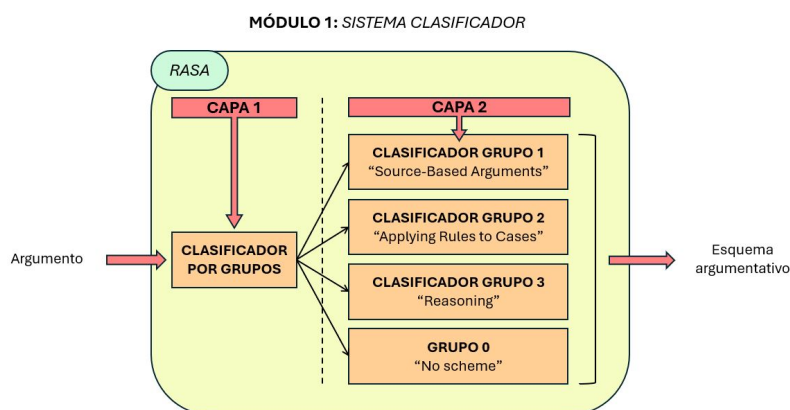


Figura 4.3: Esquema detallado del diseño del Módulo 1

- **Grupo 1:** representa a la familia de "Source-Based Arguments", la más extensa con 8 esquemas argumentativos. Todos ellos argumentos que dependen de una fuente o un agente que está en posición de conocer algo.
- **Grupo 2:** representa a la familia de "Applying Rules to Cases", con 5 esquemas argumentativos que se relacionan con una situación en la que algún tipo de regla general se aplica a los detalles de un caso específico.
- **Grupo 3:** representa a la familia de "Reasoning", con 5 esquemas argumentativos que se basan en el razonamiento para incluir diferentes tipos de secuencias en las que hay una cadena de inferencias.
- **Grupo 0:** esta categoría no representa a ninguna familia de la teoría de la argumentación, sino que incluye a todos aquellos argumentos que no siguen ningún esquema argumentativo específico.

Un ejemplo práctico del funcionamiento de este módulo sería el siguiente: si el argumento de entrada es: *"Los profesionales médicos están en posición de conocer sobre la propagación de enfermedades infecciosas en espacios públicos. Estos afirman que el uso de mascarillas en espacios públicos puede reducir significativamente la propagación de enfermedades infecciosas. Por lo tanto, usar mascarillas en espacios públicos es una medida efectiva para prevenir la propagación de enfermedades infecciosas."*, este módulo primero clasificará el argumento en un grupo, específicamente en el Grupo 1 de *"Source-Based Arguments"*. Luego, se activará el clasificador correspondiente a este grupo, clasificándolo finalmente en su esquema argumentativo: *"Argument From Position to Know (AFPK)"*.

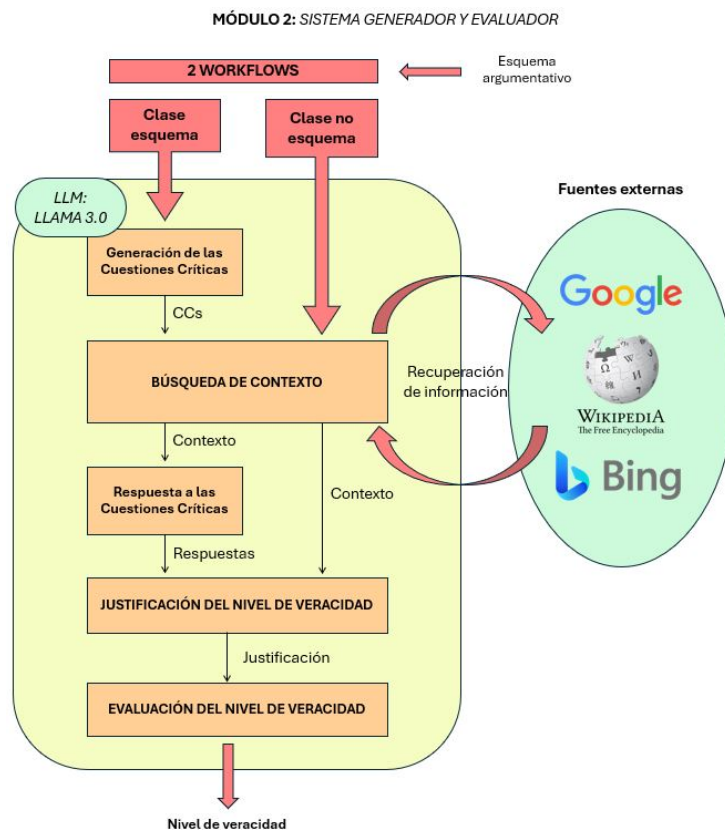
#### 4.2.2. Módulo 2: Sistema generador y evaluador del nivel de veracidad

Una vez determinado el esquema argumentativo de la frase de entrada, se puede guiar al sistema en el proceso de análisis del nivel de veracidad de esta. En este módulo se siguen una serie de pasos basados en la generación de texto y la recuperación de información, con el objetivo de devolver una puntuación (entre 0 y 1) del nivel de veracidad del argumento de entrada, donde 0 indicaría la veracidad mínima y 1 la veracidad máxima.

La estrategia utilizada es potenciar un Modelo de Lenguaje Grande (LLM) como LLaMA 3.0, proporcionándole contextualización externa de diversas fuentes de información (Google, Bing y Wikipedia) y un patrón que le permita expresar la justificación del nivel de veracidad de la frase, basándose en su esquema argumentativo y cuestiones críticas asociadas. En los capítulos 5 y 6, se explicará cómo se ha implementado esta estrategia con LLAMA en combinación con diferentes librerías para obtener el resultado final.

Por lo tanto, teniendo en cuenta la clasificación realizada en el módulo anterior, se pueden diferenciar dos flujos de trabajo en este módulo (Figura 4.4), dependiendo de si el argumento de entrada pertenece o no a un esquema argumentativo:

- **Workflow 1, Clase esquema:** Si el argumento de entrada pertenece a un esquema argumentativo, se dispone de una serie de cuestiones críticas que van dirigidas a evaluar su nivel de veracidad. Por tanto, se efectúan los siguientes pasos:
  1. Usando los patrones generales de estas cuestiones críticas, se genera el texto que las instancia, en el contexto específico del argumento a analizar.
  2. Para cada cuestión crítica se recupera información extraída de las 3 fuentes externas disponibles.
  3. Con todo este contexto, el LLM contesta a las cuestiones críticas.
  4. Basándose en estas respuestas, el LLM proporciona una justificación cualitativa del nivel de veracidad de la frase.
  5. Finalmente, el LLM evalúa esta justificación y otorga una puntuación final al argumento de entrada para expresar su nivel de veracidad.
- **Workflow 2, Clase no esquema:** Si el argumento de entrada no pertenece a ningún esquema argumentativo no se disponen de cuestiones críticas específicas. En dicho caso, buscamos directamente contexto externo en las fuentes de información a partir del propio argumento de entrada. Una vez recuperada la información necesaria, se sigue el mismo proceso que en el Workflow 1 (pasos de 4 a 5) para obtener una justificación cualitativa y un valor cuantitativo del nivel de veracidad del argumento.



Siguiendo el ejemplo práctico expuesto en el punto 4.2.1, el funcionamiento de este módulo para dicho argumento de entrada sería el siguiente:

- **Pasos 1, 2 y 3:** Tras generar las cuestiones críticas (CCs) correspondientes y recuperar información para dar respuesta a estas, el LLM llegaría a esta información:
  - **CC1:** *¿Están los profesionales médicos en posición de conocer la propagación de enfermedades infecciosas en espacios públicos? Sí, están capacitados para conocer la propagación de enfermedades infecciosas en espacios públicos debido a la efectividad de su vigilancia y cuidado en el control de brotes y la contención de patógenos.*
  - **CC2:** *¿Son los profesionales médicos fuentes honestas y fiables respecto a la propagación de enfermedades infecciosas? Aunque los profesionales médicos suelen ser confiables, también pueden generar desinformación. Por ello, es crucial apoyarse en fuentes creíbles como revistas médicas reputadas, libros de texto estándar y guías médicas.*
  - **CC3:** *¿Afirmaron los profesionales médicos que el uso de mascarillas en espacios públicos puede reducir significativamente la propagación de enfermedades infecciosas? Sí, muchos profesionales médicos y científicos han afirmado que usar mascarillas en espacios públicos puede reducir esta propagación, como lo demuestra la recomendación de Hong Kong de usar mascarillas para prevenir virus respiratorios.*
- **Paso 4:** el LLM proporcionaría la siguiente justificación del nivel de veracidad del argumento: *Basado en estas respuestas, el nivel de veracidad de esta frase es alto. El argumento es sólido porque los profesionales médicos y científicos están en posición de conocer la propagación de enfermedades infecciosas, han afirmado que el uso de mascarillas puede reducir la propagación, y hay evidencia de fuentes creíbles que respaldan esta afirmación. Sin embargo, se debe tener en cuenta el contexto en el que se ha usado este argumento y la posibilidad de desinformación o sesgo por parte de ciertas fuentes.*

- **Paso 5:** Finalmente, se obtiene la puntuación final del nivel de veracidad de la frase: **0.8/1.0**

### 4.2.3. Interfaz web del sistema

La solución implementada para el sistema de detección de desinformación se presenta en formato de herramienta web. Dicha herramienta presenta una interfaz sencilla, de fácil uso e intuitiva donde el usuario debe introducir el argumento en inglés que quiere evaluar y presionar el botón *EVALUAR* para que el sistema inicie el procesamiento. De este modo, se activará todo el backend del sistema detector de desinformación con la ejecución de cada uno de sus módulos conectados. Finalmente, se visualizarán por pantalla los resultados obtenidos: el esquema argumentativo al que pertenece el argumento de entrada, la justificación cualitativa y el valor cuantitativo del nivel de veracidad, así como una recopilación de los enlaces a las fuentes que se han consultado. En la Figura 4.5, se muestra un *mockup* básico y esquemático de la interfaz de la herramienta web desarrollada.

El mockup muestra una interfaz web con un fondo verde claro. En la parte superior, un recuadro con el título "SISTEMA DETECTOR DE DESINFORMACIÓN". Debajo, el texto "Introduce el argumento a evaluar:" precede a un campo de entrada de texto naranja y un botón rojo con el texto "EVALUAR". A continuación, el texto "Justificación:" precede a un campo de entrada de texto gris. Debajo de este campo, el texto "Score" precede a un campo de entrada de texto gris. Finalmente, el texto "Enlaces a fuentes consultadas :" precede a un campo de entrada de texto gris.

Figura 4.5: Diseño visual y estructurado del sitio web

## 4.3 Tecnología utilizada

En esta sección se presentan todas las tecnologías, herramientas y frameworks que se han utilizado en el desarrollo del sistema de detección de desinformación. Posteriormente, en el capítulo 5 se explicará más detalladamente el procedimiento seguido y cómo se han implementado las soluciones concretas.

### 4.3.1. Rasa

Rasa<sup>1</sup> es un framework de código abierto y altamente extensible para el aprendizaje automático, que permite desarrollar sistemas conversacionales tanto por texto como por voz a través de chatbots. Esta librería ofrece una automatización altamente personalizable de las interacciones entre el ordenador y el usuario. De este modo, se facilita la

<sup>1</sup><https://rasa.com/>



implementación de chatbots adaptados a las necesidades, datos y requisitos específicos de cada usuario.

El framework de Rasa está compuesto por tres módulos diferenciados. En primer lugar, el módulo dedicado al procesamiento del lenguaje natural (*Rasa NLU*) que se enfoca en la extracción de entidades y la detección de intenciones. En segundo lugar, otro módulo dirigido a la gestión de diálogos (*Rasa Core*) y, en tercer y último lugar, un módulo para integraciones con otros sistemas permitiendo su uso en diversas plataformas, como sitios web y redes sociales.

En este trabajo, utilizamos principalmente la funcionalidad de Rasa NLU para construir el sistema de clasificadores del Módulo 1, siguiendo una estrategia similar a la propuesta en [de Sousa et al., 2024]. A partir de un entrenamiento de la unidad de PLN de Rasa con un dataset de argumentos escritos en inglés, se consigue realizar una clasificación de argumentos según su esquema argumentativo.

### 4.3.2. LLAMA

LLAMA<sup>2</sup> (Large Language Model Meta AI) es un modelo de lenguaje grande (LLM) de código abierto desarrollado por Meta AI. Este modelo ha demostrado ser capaz de realizar una amplia variedad de tareas de procesamiento del lenguaje natural, ganando popularidad en los últimos años debido a su versatilidad y potencia.

Comparándolo con sus competidores de código propietario, Llama destaca en aspectos como: disponibilidad de código abierto, técnicas avanzadas de entrenamiento, una sólida precisión de los datos y una arquitectura optimizada que permite el uso eficiente de recursos computacionales. Aunque modelos convencionales como GPT-3 y GPT-4 de OpenAI, PaLM y Gemini de Google, y Claude de Anthropic también demuestran una gran capacidad de comprensión, pueden evidenciar problemas de recursos computacionales, eficiencia y costo, ya que superan en gran medida los miles de millones de parámetros de los modelos más grandes disponibles en LLAMA.

Concretamente, se ha investigado la familia de LLMs de LLAMA 2 y LLAMA 3, que fueron publicados en 2023 y el 18 de abril de 2024, respectivamente. Los modelos de Llama 2 están disponibles en tres configuraciones diferentes: siete mil millones (7B), 13 mil millones (13B) o 70 mil millones de parámetros (70B). Mientras que Llama 3 se ofrece también en 3 tamaños diferentes: 7B, 80B y 400B (este último a fecha de hoy no está disponible).

En el contexto de este trabajo, se han analizado y realizado pruebas (ver capítulo 6) con LLAMA 2 13B y LLAMA 3 70B, y finalmente se ha optado por utilizar LLAMA 3 70B en el Módulo 2 para desarrollar el sistema generador y evaluador del nivel de veracidad de los argumentos.

### 4.3.3. LangChain

LangChain<sup>3</sup> es un framework de código abierto, disponible en Python y Javascript, diseñado para facilitar y potenciar el desarrollo de aplicaciones que hacen uso de LLMs, como chatbots y asistentes virtuales. Su característica principal es el nivel de centralización y abstracción que presenta, ofreciendo componentes modulares, funciones y clases de objetos que se pueden combinar fácilmente para programar aplicaciones de IA generativa. Sus componentes incluyen plantillas de prompts, cadenas de flujo, agentes que

---

<sup>2</sup><https://llama.meta.com/>

<sup>3</sup><https://www.langchain.com/>

actúan como motores de inferencia, acceso a índices de carga de documentos o bases de datos vectoriales, guardado de la memoria del LLM y conexiones a otras herramientas.

Aunque este enfoque abstracto puede limitar el nivel de personalización de los programas en desarrollo, esta biblioteca es muy útil para la experimentación y creación rápida de prototipos. En este trabajo, se ha experimentado combinando LangChain con LLAMA 2 13B en el sistema generador del Módulo 2. Sin embargo, en la versión final del sistema de detección de desinformación no se utiliza esta librería y se opta por desarrollar una versión más personalizable por el programador. En el capítulo 6, se describen las pruebas experimentales realizadas y se justifica más detalladamente la alternativa final seleccionada.

#### 4.3.4. Otras librerías de Python

##### Recuperación de información

Durante la etapa de recuperación de información de diversas fuentes se utilizan las librerías:

- **SerpApi<sup>4</sup> y su módulo Google Search Results**: una API que permite a los desarrolladores recuperar fácilmente y de forma eficiente información disponible en Google y otros motores de búsqueda importantes. Concretamente, en este proyecto, se ha utilizado una versión de esta API que permite 100 consultas gratuitas al mes para acceder a consultas de búsqueda en los motores de Google y Bing.
- **Wikipedia API<sup>5</sup>**: esta librería admite la extracción de textos, secciones, enlaces y otras categorías de las diversas páginas de Wikipedia.

##### Paralelización

En la fase de recuperación del Módulo 2 se ha llevado a cabo una paralelización de las tareas de búsqueda de información a través de la librería:

- **concurrent.futures<sup>6</sup>**: un módulo de Python que proporciona una interfaz de alto nivel para la ejecución asíncrona de tareas, sin bloquear el flujo principal del programa.

##### Aplicación web

El acceso al sistema de detección de desinformación se realiza a través de una herramienta web, cuya implementación y diseño se ha desarrollado con el siguiente framework:

- **Streamlit<sup>7</sup>**: librería de Python que permite la creación de aplicaciones web interactivas y facilita el desarrollo de aplicaciones de aprendizaje automático con visualización de datos. Además, dispone de la posibilidad de integrar código HTML y CSS para personalizar aún más su interfaz de usuario.

---

<sup>4</sup><https://serpapi.com/>

<sup>5</sup>[https://www.mediawiki.org/wiki/API:REST\\_API/es](https://www.mediawiki.org/wiki/API:REST_API/es)

<sup>6</sup><https://docs.python.org/3/library/concurrent.futures.html>

<sup>7</sup><https://streamlit.io/>

#### 4.3.5. Entornos y sistema operativo

Toda la programación del sistema se ha realizado en el entorno de **Windows 10**, aprovechando su compatibilidad con las herramientas y tecnologías necesarias. El lenguaje de programación principal de desarrollo ha sido **Python**, integrándose además fragmentos de código HTML y CSS para personalizar aspectos visuales de la interfaz en la parte del desarrollo web. Como editor de código principal, se ha utilizado **Visual Studio Code (VSC)**, cuya versatilidad y amplia gama de extensiones permiten una gestión clara y eficiente del código. También se han realizado pruebas en **Google Colab**, ya que ofrece un entorno de ejecución de Jupyter Notebooks en la nube y acceso a recursos de hardware acelerados, altamente necesarios para la carga del modelo de LLAMA 2 durante las pruebas experimentales con LangChain.



---

---

## CAPÍTULO 5

# Desarrollo de la solución propuesta

---

Este capítulo detalla el desarrollo del sistema de detección de desinformación, describiendo las decisiones tomadas y los matices presentes en la implementación de cada uno de los módulos que integran el sistema final. En la sección 5.1 se aborda el objetivo OE2 mediante el desarrollo del sistema clasificador en esquemas argumentativos. Seguidamente, en la sección 5.2 se abordan los objetivos EO3, EO4 y EO6, a través del desarrollo de un sistema evaluador del nivel de veracidad de argumentos con contextualización externa optimizado con una paralelización de tareas utilizando programación concurrente. Finalmente, en la sección 5.4 se explica el desarrollo de la interfaz web del sistema abordando el objetivo EO5. Todo el código del proyecto desarrollado se encuentra disponible bajo demanda en el siguiente repositorio<sup>1</sup>.

## 5.1 Módulo 1: Sistema clasificador en esquemas argumentativos

---

### 5.1.1. Dataset utilizado

El conjunto de datos utilizado en el sistema clasificador del módulo 1 ha sido constituido a partir del dataset "NLAS-MULTI corpus" [Ruiz-Dolz et al., 2024]. Este dataset original<sup>2</sup> contiene argumentos en inglés y en español a favor o en contra de 50 temas que han sido adaptados a 20 esquemas argumentativos de la teoría de la argumentación de Walton especificando sus premisas y conclusiones en formato JSON. Es uno de los corpus más destacados y de mayor tamaño actualmente disponibles para tareas de "Argument Mining", ya que dispone de 3.810 argumentos automáticamente generados a partir de las APIs de GPT-3.5-TURBO y GPT-4, y validados por validadores humanos expertos en argumentación.

Se han implementado varias modificaciones a este dataset base para construir el dataset específico utilizado en este proyecto. En primer lugar, se han seleccionado los argumentos en inglés, descartando aquellos disponibles en español, ya que nos hemos centrado en desarrollar el sistema detector de desinformación en el idioma inglés, dominante en el ámbito del "Argument Mining". En segundo lugar, de los 20 esquemas argumentativos originales del dataset base, se han descartado dos: el Argumento de la Mejor Explicación y el Argumento de la Pendiente Resbaladiza. El primero fue eliminado porque no sigue un esquema argumentativo claramente diferenciado y único, pudiendo asociarse a otros esquemas argumentativos más generales, mientras que el segundo, fue descartado debido a la existencia de muchas variantes diferentes, lo que dificulta la concreción en una variante principal.

---

<sup>1</sup>Código del proyecto en repositorio privado de: <https://github.com/anagutierr>

<sup>2</sup><https://zenodo.org/records/8364002>

Por otra parte, se ha identificado la necesidad de incluir una nueva clase en el dataset para clasificar los argumentos que no siguen ningún esquema argumentativo específico. La creación de esta nueva clase es esencial, ya que a menudo los argumentos no suelen presentarse siguiendo una estructura argumentativa definida, sino que se exponen de manera más informal y natural. De este modo, se ajusta y refina el sistema para funcionar en escenarios más reales y cotidianos.

La inclusión de esta clase "no esquema" ha supuesto un desafío importante, ya que al ser una clase genérica, podría ser causante de la inclusión de bastante ruido en el modelo. Este problema se ha resuelto en gran medida al dotar al sistema clasificador de 2 capas o fases: una primera capa donde los argumentos se separan por grupos argumentativos (incluyendo el "grupo0" que representa la clase "no esquema"), y una segunda capa donde se clasifican según el esquema argumentativo específico. Más adelante, en el Capítulo 6, se explicarán y expondrán los resultados de los experimentos realizados en este módulo. Inicialmente, se probó la opción de un sistema clasificador unicapa, pero posteriormente se implementó una versión bicapa que, al obtener mejores resultados, fue por la que se optó en el sistema final.

Se ha seguido la misma estrategia del "NLAS-MULTI corpus" para generar automáticamente ejemplos que pertenezcan a la clase "no esquema", utilizando en este caso la API de GPT 3.5. Además, es importante destacar que en el resto de esquemas argumentativos también se han generado automáticamente algunos argumentos adicionales que cumpliendo con sus respectivos esquemas, siguen una estructura más variada. De esta forma, se obtiene un mayor número de datos así como una mayor variedad y generalización del modelo.

Como resultado final, nuestro dataset definitivo está formado por 2,188 argumentos en inglés tanto a favor como en contra de los 50 temas originales, y asociados a 19 esquemas argumentativos. En la Figura 5.1, se pueden observar algunos de los temas tratados en estos argumentos y la distribución completa del dataset final: con sus grupos y esquemas argumentativos.

TEMAS TRATADOS	
• Eutanasia	
• Vacunación obligatoria	
• Apariencia física en el éxito	
• Experimentación animal	
• Cambio climático	
• Legalización del cannabis	
• Aborto	
• Libertad de expresión	
• Clonación animal	
• Investigación en inteligencia artificial	
• Energía nuclear	
• Uso de redes sociales	
• Control de armas	
• Cuotas de género	
• Terraplanismo	
• Energía renovable	
• Transporte eléctrico	
• Coches totalmente autónomos	
• Inmigración	
• ...	

Nº total de muestras		Total
GRUPO 0: no_scheme		321
GRUPO 1: Source-Based Arguments		828
GRUPO 2: Applying Rules to Cases		528
GRUPO 3: Reasoning		511

GRUPO 0	Total
no_scheme	321

GRUPO 1	Total
direct_ad_hominem	99
expert_opinion	110
ignorance	103
inconsistent_commitment	98
popular_opinion	99
popular_practice	99
position_to_know	110
witness_testimony	110

GRUPO 2	Total
analogy	107
established_rule	105
example	107
precedent	104
verbal_classification	105

GRUPO 3	Total
cause_to_effect	108
sign	110
sunk_costs	102
threat	95
waste	96

Figura 5.1: Recopilación de algunos temas tratados y distribución completa del dataset utilizado

Por último, cabe destacar que el conjunto de datos utilizado en este proyecto está en formato YAML (YAML Ain't Markup Language) [Ben-Kiki et al., 2009], debido a que es el formato estándar utilizado por Rasa (tecnología empleada en este módulo para la implementación del sistema clasificador). El dataset se ha integrado y almacenado en

los diferentes archivos de datos YAML del sistema de archivos local del proyecto. Este formato de serialización de datos es conocido por ser bastante legible y seguir una estructura jerárquica con indentación. Por ejemplo, en la Figura 5.2 se observa un ejemplo de un archivo YAML del proyecto en el que se puede apreciar el formato que siguen los datos. Contiene diversos argumentos etiquetados según el *intent* del grupo 0, indicando por tanto, que pertenecen a dicha clase. En el siguiente apartado, se explica más detalladamente cómo se ha utilizado este formato para llevar a cabo el entrenamiento del sistema con Rasa.

```
- intent: grupo0
  examples: |
    - Proponents argue that euthanasia provides a compassionate end for those suffering,
      while opponents emphasize the sanctity of life.
    - Mandatory vaccination during pandemics is viewed as a crucial public health measure
      to curb the spread of diseases and protect vulnerable populations.
    - The emphasis on physical appearance for personal success perpetuates harmful
      stereotypes and undermines individual worth based on merit.
    ...
```

Figura 5.2: Ejemplo de un archivo YAML del sistema en el que se puede apreciar el formato que siguen los datos

### 5.1.2. Entrenamiento con Rasa

Como se explicó con anterioridad en el Capítulo 4, se ha utilizado Rasa para implementar el sistema clasificador del módulo 1. Rasa es un framework orientado a la creación de asistentes y chatbots, sin embargo, en este trabajo no se ha empleado con ese propósito. En su lugar, se ha aprovechado la lógica y el funcionamiento que Rasa utiliza para crear chatbots, con el objetivo de construir los 4 clasificadores que conforman el sistema clasificador bicapa de este módulo.

Inicialmente, en un entorno virtual de Anaconda previamente configurado en nuestro equipo, se descargó e instaló la *versión 3.6.20 de Rasa*, la más reciente disponible en ese momento. Posteriormente, utilizando el comando "*rasa init*", se creó un proyecto base inicial sobre el que partir y aplicar las modificaciones pertinentes para ajustarlo a nuestra tarea. Este proyecto base de Rasa consistía en un chatbot llamado *Moodbot*, con la capacidad de saludar y preguntar por el estado de ánimo del usuario ejecutando diferentes acciones según la respuesta del usuario.

Los chatbots de Rasa trabajan con intenciones o "*intents*", que representan los propósitos del usuario y que el chatbot debe de ser capaz de detectar para actuar en consecuencia. Por ejemplo, en la Figura 5.3, se muestran varios ejemplos de texto, como el mensaje en inglés "*good morning*", que tienen la intención de saludar ("*greet*"). Proporcionando múltiples ejemplos para cada intención que se haya definido, el asistente será capaz de clasificar cualquier entrada o mensaje en su correspondiente intención de manera precisa.

```
nlu:
  - intent: greet
  - examples: |
    - hey
    - hello
    - hi
    - hello there
    - good morning
    - good evening
```

Figura 5.3: Ejemplo de la intención "*greet*" definida en Rasa en la que se proporcionan varios ejemplos de frases que se utilizan para saludar

Tal y como se ha presentado en el Capítulo 4, nuestro sistema clasificador está diseñado en 2 capas y contiene 4 clasificadores en total.

- En la **primera capa**, se implementa un asistente clasificador de Rasa cuyas intenciones se corresponden con los 4 grupos argumentativos definidos previamente.
- En la **segunda capa**, hay tres clasificadores más. Para cada grupo argumentativo (a excepción del grupo 0), se implementa un asistente clasificador cuyas intenciones son los esquemas argumentativos asociados a ese grupo. No existe clasificador para el grupo 0, ya que si un argumento es clasificado por la primera capa en este grupo significa que no pertenece a ningún esquema argumentativo incluyéndose en la clase "no esquema".

Gracias al desarrollo de este sistema, cuando se recibe un argumento de entrada, el asistente clasificador de la primera capa identificará a qué intención (*grupo argumentativo*) pertenece y, seguidamente, se activará el asistente clasificador de ese grupo, el cual detectará de la misma forma a qué intención (*esquema argumentativo*) pertenece el argumento.

Por tanto, se han creado 4 asistentes básicos y se han modificado los diferentes archivos de cada uno para convertirlos en asistentes clasificadores de grupos o esquemas argumentativos según la capa en la que se encuentren. Los archivos más importantes que forman cada asistente clasificador son:

- "**domain.yml**": en este archivo se registran todas las *clases* o "*intents*" que queremos que nuestro clasificador sea capaz de detectar. Por ejemplo, para el clasificador de la capa 1 se especifican como intenciones (Figura 5.4) los grupos argumentativos definidos previamente.

```
intents:
- grupo1 #Source-Based Arguments
- grupo2 #Applying Rules to Cases
- grupo3 #Reasoning
- grupo0 #No scheme

responses:
utter_grupo0:
- text: "Grupo0"

utter_grupo1:
- text: "Grupo1"

utter_grupo2:
- text: "Grupo2"

utter_grupo3:
- text: "Grupo3"
```

**Figura 5.4:** Intenciones definidas en el archivo "*domain.yml*" del clasificador en grupos argumentativos de la capa 1

- "**nlu.yml**": este archivo es el más extenso ya que guarda, para cada "*intent*" definido en "*domain.yml*", un conjunto de frases ejemplo que pertenecen a esa *clase* o intención y que el bot utilizará en su entrenamiento. Por tanto, en este archivo se almacenan todos los argumentos del dataset organizados según la clase a la que pertenecen. En la Figura 5.5, podemos observar algunos ejemplos de argumentos que pertenecen a las 4 intenciones (*grupos argumentativos*) definidas en el asistente clasificador de la capa 1.
- "**stories.yml**": en este archivo se registran las conversaciones entre el bot y los usuarios que se utilizarán para entrenar al asistente. Cada "*story*" permite al asistente



```

- intent: grupo0
  examples: |
    - Proponents argue that euthanasia provides a compassionate end for those suffering,
      while opponents emphasize the sanctity of life.
    - Mandatory vaccination during pandemics is viewed as a crucial public health measure
      to curb the spread of diseases and protect vulnerable populations.
    - The emphasis on physical appearance for personal success perpetuates harmful
      stereotypes and undermines individual worth based on merit.
    ...
- intent: grupo1
  examples: |
    - "{\n \"major premise\": \"Medical professionals (such as doctors and nurses) are in position to know
      about the treatment options available for terminally ill patients.\",\n \"minor premise\": \"Many
      medical professionals argue that euthanasia is a humane option for terminally ill patients who are
      experiencing unbearable suffering and have little hope for recovery.\",\n \"conclusion\": \"Euthanasia
      can be a morally justifiable option for terminally ill patients who are experiencing unbearable
      suffering and have little hope for recovery.\""}"
    ...
- intent: grupo2
  examples: |
    - "{\n \"similarity premise\": \"Generally, when a pet is suffering and has no chance of recovery,
      it is often considered humane to euthanize them to end their suffering\",\n \"base premise\": \"If a
      human is suffering and has no chance of recovery, it should be just as humane to allow them the choice
      of euthanasia to end their suffering\",\n \"conclusion\": \"Therefore, if euthanasia is considered
      humane for pets, it should also be considered humane for humans who are suffering and have no chance
      of recovery\""}"
    ...
- intent: grupo3
  examples: |
    - "{\n 'major premise': 'Generally, if a person is terminally ill and suffering greatly, continued
      medical treatment may not alleviate their pain or improve their quality of life',\n 'minor premise':
      'In this case, a person is terminally ill and suffering greatly',\n 'conclusion': 'Therefore, in this
      case, allowing euthanasia as an option may alleviate their pain and end their suffering'}"
    ...

```

Figura 5.5: Recopilación de algunos ejemplos especificados para cada intención en el archivo "nlu.yml" del clasificador de la capa 1

elegir la mejor acción en respuesta a cada interacción con el usuario. Dado que nuestros asistentes funcionan como clasificadores, no se requiere definir interacciones complejas entre el usuario y el asistente. Por tanto, en este archivo se definen "stories" simples, con una sola interacción que representa la clasificación directa de la frase.

- "*config.yml*": este es el documento de configuración donde se define el *pipeline* y los parámetros que se utilizarán para analizar y clasificar el texto de entrada del asistente. La configuración empleada en el módulo *Rasa NLU* es la misma para los 4 clasificadores. Concretamente, el *pipeline* utilizado está formado por los siguientes componentes<sup>3</sup>:
  - *Tokenizers*: se utiliza el *WhitespaceTokenizer* [Vijayarani et al., 2016] para dividir el texto de entrada en *tokens* basándose en los espacios en blanco de la cadena.
  - *Featurizers*: son extractores de características que facilitan al modelo la comprensión del significado del texto de entrada. Generan características a partir de texto en forma de representaciones numéricas. Los extractores utilizados han sido: el *RegexFeaturizer* [Hwang et al., 2021] que genera características basadas en patrones regulares, el *LexicalSyntacticFeaturizer* [Mishra et al., 2022] que se basa en el análisis léxico y sintáctico del texto, y el *CountVectorsFeaturizer* [Nguyen et al., 2021] que captura la frecuencia de las palabras convirtiéndolas en vectores de recuento. Además, se ha aplicado otro componente

<sup>3</sup>Documentación componentes RASA: <https://rasa.com/docs/rasa/components/>

conocido como *EntitySynonymMapper* [Shabbir et al., 2021], el cual es capaz de mapear sinónimos de entidades a la entidad o palabra principal.

- **Classifiers:** son los componentes encargados de determinar a qué "intents" pertenecen los argumentos de entrada. Se emplean los siguientes clasificadores: el *DIETClassifier* [Astuti et al., 2021], un clasificador basado en la arquitectura *Transformer* que utiliza *embeddings* de texto bidireccionales para predecir la intención y las entidades, el *ResponseSelector* [Kong et al., 2021] que selecciona respuestas predefinidas basadas en el contexto del mensaje, y, por último, el *FallbackClassifier* [Khan et al., 2021] que se encarga de identificar mensajes que no se pueden clasificar con suficiente confianza. El umbral de confianza mínimo para que el clasificador principal considere una predicción válida se establece en 0,3 sobre 1. Además, el entrenamiento de los componentes *DIETClassifier* y *ResponseSelector* se realiza durante 100 épocas, aplicando una restricción de similitud *constrain\_similarities* para mejorar la capacidad de generalización.

Después de definir toda esta información en el sistema de archivos, se realiza una partición del dataset con el comando "*rasa data split nlu*", dedicando aproximadamente el 80 % a entrenamiento y el 20 % a test. Esta técnica de partición es conocida como *hold-out method*, donde se separa el conjunto de datos disponible en dos particiones independientes: una dedicada al entrenamiento del modelo y la otra dedicada a su evaluación, asegurando que los datos de prueba no sean conocidos por el modelo durante la fase de entrenamiento.

El procedimiento de entrenamiento de los clasificadores se lleva a cabo utilizando el comando "*rasa train nlu -nlu train\_test\_split/training\_data.yml*", que entrena los modelos especificados en el archivo *config.yml* utilizando el conjunto de datos de entrenamiento. Los modelos se entrenan durante 100 épocas, aplicando la restricción de similitud *constrain\_similarities* para mejorar la capacidad de generalización. Mediante aprendizaje supervisado, se optimizan y ajustan iterativamente los pesos de los clasificadores definidos en el *pipeline* anterior con el objetivo de reducir el error de predicción en el conjunto de entrenamiento.

Posteriormente, se evalúan los modelos clasificadores entrenados utilizando los datos de test. Mediante el comando "*rasa test nlu -nlu train\_test\_split/test\_data.yml*" se mide el rendimiento de los modelos, obteniendo métricas de evaluación como: precisión, F1-Score, Recall y Accuracy. Estos resultados, se presentan en el Capítulo 6.

Es importante destacar que, durante la etapa de entrenamiento los argumentos seguían una estructura con las premisas y conclusiones claramente marcadas en cada uno (como se aprecia en los grupos 1, 2 y 3 de la Figura 5.5). Sin embargo, en el conjunto de test, estas marcas se han eliminado, dejando todos los argumentos en lenguaje natural con el objetivo de adaptar las pruebas a un escenario real. En el conjunto de entrenamiento se ha mantenido este formato estructurado y marcado, ya que, según se demuestra en [de Sousa et al., 2024], esta aproximación proporciona un mayor aprendizaje y confianza en los resultados de los modelos implementados en Rasa.

## 5.2 Módulo 2: Sistema generador y evaluador del nivel de veracidad

---

### 5.2.1. Configuración y uso de LLAMA

En este módulo se utiliza un LLM de la familia de LLAMA con contextualización externa a partir de diversas fuentes de información. Aunque inicialmente se probó con otra

versión de LLAMA siguiendo una estrategia diferente (detallada como un experimento en el Capítulo 6), en la versión final se ha optado por utilizar un modelo de LLAMA a través de una API REST del servicio PoliGPT<sup>4</sup> de la *Universitat Politècnica de València* (UPV).

Actualmente, PoliGPT es un servicio innovador de inteligencia artificial en fase de pruebas, gestionado por el Área de Sistemas de Información y Comunicaciones de la UPV. Está basado en Ollama<sup>5</sup>, una herramienta diseñada para ejecutar modelos de lenguaje en máquinas locales y su objetivo es proporcionar acceso a modelos de lenguaje de gran tamaño de manera sencilla, sin necesidad de una infraestructura en la nube o de ordenadores con gran capacidad computacional. Aunque PoliGPT tiene su propio chatbot, en este proyecto únicamente se ha utilizado su API REST debido a su capacidad para integrar de manera directa y eficiente los grandes modelos de lenguaje disponibles en PoliGPT con el sistema desarrollado.

La API REST de PoliGPT es compatible con la API de OpenAI, facilitando de este modo el uso de varios modelos de lenguaje a través de diferentes lenguajes de programación y su posterior integración en múltiples aplicaciones. La integración con Ollama permite enviar solicitudes a los modelos de lenguaje que se ejecutan localmente en los sistemas de la UPV. Para aprovechar esta funcionalidad, ha sido necesario obtener un *endpoint*<sup>6</sup> y una clave de acceso que permitiera utilizar la API REST de este servicio.

En este proyecto se ha utilizado el modelo de PoliGPT denominado *poligpt:latest*, basado en *llama3:70b-instruct-q6\_K*<sup>7</sup>. Este modelo pertenece a la familia de modelos LLAMA, explicada previamente en el Capítulo 4, y destaca en el campo del procesamiento de lenguaje natural por ser uno de los pocos LLM abiertamente disponible y más potente en la actualidad. Está compuesto por 70 mil millones de parámetros y ha sido refinado específicamente para seguir instrucciones, mejorando su capacidad para comprender y ejecutar comandos requeridos por los usuarios. En particular, se utiliza su versión cuantizada (sufijo *-q6\_K*), que reduce su tamaño y mejora su eficiencia en términos de memoria y velocidad, sin sacrificar significativamente la precisión y el rendimiento.

Una vez configurado el servicio descrito anteriormente, el proceso seguido para realizar las interacciones con el modelo ha sido el siguiente:

1. **Creación del cliente:** se ha configurado un cliente para la API REST utilizando la librería OpenAI de Python. Este cliente se inicializa con el *endpoint* y la clave API mencionados anteriormente, permitiendo que las solicitudes sean enviadas y procesadas a través de Ollama.
2. **Envío de solicitudes:** una vez inicializado el cliente, se pueden enviar y procesar las solicitudes (*chat completions*) al modelo alojado en los sistemas de la UPV. Estas solicitudes incluyen una serie de mensajes en formato JSON que proporcionan instrucciones al modelo sobre la tarea a realizar. Adicionalmente se configuran los siguientes parámetros:
  - **Role:** se utiliza para definir el contexto y la perspectiva desde la cual el modelo debe interpretar y responder a las entradas del usuario. Se utilizan dos roles: el *system*, para establecer el comportamiento general del modelo y las reglas que debe seguir durante la interacción, y el *user* para representar las entradas específicas del usuario o las preguntas directas dirigidas al modelo.

<sup>4</sup><https://wiki.upv.es/confluence/pages/viewpage.action?pageId=974159959>

<sup>5</sup><https://www.ollama.com/>

<sup>6</sup><https://poligpt.upv.es/api/desinforma/ollama>

<sup>7</sup>[https://ollama.com/library/llama3:70b-instruct-q6\\_K](https://ollama.com/library/llama3:70b-instruct-q6_K)

- **Response Format:** define el formato en el que se espera que el modelo devuelva su respuesta. En la mayoría de solicitudes, se ha establecido un formato de respuesta en JSON, facilitando así la integración posterior de esta información en el flujo de trabajo del sistema.

El sistema generador y evaluador del módulo 2 tiene como objetivo proporcionar una justificación cualitativa y cuantitativa del nivel de veracidad del argumento de entrada, apoyándose en el contexto recuperado y el razonamiento seguido en el esquema argumentativo detectado en el módulo 1. Para ello, se llevan a cabo una serie de solicitudes de *chat completion* secuenciadas de la siguiente manera (Figura 5.6):

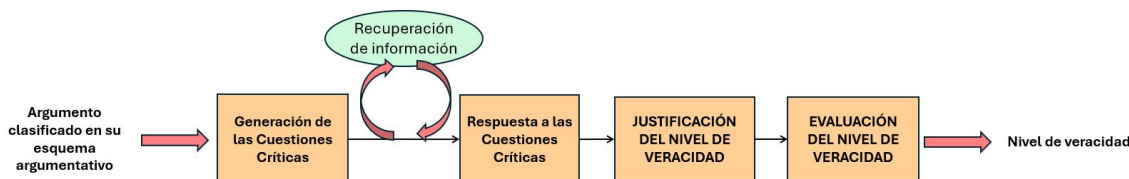


Figura 5.6: Flujo de las solicitudes de *chat completion* realizadas al modelo

1. **Generación de las cuestiones críticas:** dado el argumento de entrada, la definición del esquema argumentativo que cumple y la definición de las cuestiones críticas asociadas a este esquema, se le solicita al modelo que personalice estas cuestiones según el contenido del argumento de entrada y las presente en formato JSON.
2. **Respuesta a las cuestiones críticas:** tras recuperar información de fuentes externas, se obtiene un contexto que se envía junto con las cuestiones críticas definidas. Con ello, se le solicita al sistema que responda a estas cuestiones basándose en el contexto proporcionado y presentando las respuestas en formato JSON.
3. **Justificación del nivel de veracidad:** esta es la solicitud más extensa. Partiendo de toda la información recopilada previamente por el sistema (el argumento de entrada, el esquema argumentativo que cumple, las cuestiones críticas y las respuestas a estas cuestiones), se le solicita al sistema que actúe como un asistente experto en argumentación computacional, obteniendo una justificación estructurada sobre el nivel de veracidad del argumento de entrada.
4. **Evaluación del nivel de veracidad:** finalmente, dada la justificación estructurada proporcionada por el sistema, se le solicita que estime un valor numérico entre 0 y 1 que refleje cuantitativamente el nivel de veracidad del argumento evaluado. Siendo 0 la estimación más baja de veracidad y 1 la más alta.

### 5.2.2. Recuperación de información

El objetivo de la fase de recuperación de información es ajustar y mejorar el rendimiento del LLM de LLAMA mediante la contextualización externa proporcionada por diversas fuentes de información: Google, Bing y Wikipedia. En la sección 4.3, se explican las bibliotecas utilizadas para extraer información de estas fuentes: SerpAPI, que permite la conexión con Google Search y Bing a través de una clave de acceso, y la API de Wikipedia.

El proceso para recuperar información externa se inicia tras la obtención de las cuestiones críticas. Para cada cuestión crítica, se realiza una búsqueda en cada una de las 3 fuentes de información consideradas. Es decir, para cada cuestión crítica se llevan a cabo

en total 3 búsquedas, una para cada fuente. Los resultados de estas búsquedas se almacenan en un diccionario Python.

Al realizar las búsquedas, el sistema guarda los fragmentos de texto (*snippets* de Google y Bing), resúmenes de Wikipedia y los enlaces URL de los resultados. Los parámetros específicos de búsqueda que se han definido son:

- **Ubicación:** establecida en Estados Unidos. Esta ubicación permite obtener resultados más acordes con el contexto del dataset del proyecto, teniendo en cuenta además que la mayoría de los recursos de información en dicha ubicación están disponibles en inglés.
- **Idioma:** configurado en inglés. El sistema funciona en este idioma.
- **Número de páginas visitadas en cada búsqueda:** el valor por defecto inicial se estableció en 6 páginas, pero debido al límite de consultas gratuitas de SerpAPI, se decidió reducir el número a una. Aunque aumentar el número de páginas puede proporcionar mayor contextualización al modelo, también puede ser contraproducente. A mayor contexto suministrado, el modelo puede perder el foco de su tarea, desviando su atención y provocando alucinaciones que lleven a engaños. En este caso, reducir el número de visitas a uno no resulta insuficiente, ya que hay que tener en cuenta que adicionalmente y de manera independiente se realizan otras dos búsquedas en el resto de fuentes.

Habría que resaltar también que la búsqueda en Wikipedia podría arrojar errores, como una página *wiki* no encontrada o un error de desambiguación ocasionado cuando la consulta proporciona páginas *wiki* ambiguas. No obstante, es importante destacar que estos errores se manejan adecuadamente para garantizar que se obtenga al menos un resumen y una URL válidos por cada cuestión crítica.

Inicialmente, la fase de recuperación de información se implementó con búsquedas secuenciales, lo que provocaba un funcionamiento más lento del sistema. Pero, posteriormente, se llevó a cabo un proceso de paralelización que optimizó notablemente la velocidad y la eficiencia de este módulo. En el capítulo 6, se compara el tiempo de ejecución entre el sistema basado en procesamiento secuencial y el optimizado con procesamiento paralelo.

Esta paralelización se realiza utilizando la clase *ThreadPoolExecutor* del módulo *concurrent.futures* de Python, permitiendo ejecutar múltiples búsquedas simultáneamente en las fuentes de información (Google, Bing y Wikipedia). Además, para cada fuente de información, también se realizan múltiples búsquedas en paralelo, ya que por cada cuestión crítica que tenga el esquema se realiza una búsqueda en esa fuente específica. Por ejemplo, si un esquema argumentativo tiene 3 cuestiones críticas, se realizan un total de 9 búsquedas en paralelo (3 búsquedas en cada fuente).

---

## 5.3 Interconexión de los módulos

---

El sistema de archivos del proyecto está organizado en los siguientes componentes:

- **Carpeta "*ClasificadorCAPAI*":** donde está guardado el modelo clasificador en grupos argumentativos de la capa 1. Contiene el modelo entrenado en un archivo comprimido *.tar.gz* con los pesos y parámetros correspondientes.

- **Carpeta "ClasificadoresCAPA2"**: donde se guardan los tres modelos clasificadores en esquema argumentativos de la capa 2. Contiene tres archivos comprimidos con los pesos de cada modelo entrenado.
- **"schemes.json"**: archivo JSON donde se almacena la información de los 19 esquemas argumentativos (nombre del esquema, reglas del esquema y sus cuestiones críticas). Funciona como un archivo de datos externo al que el sistema accede una vez ha clasificado el argumento de entrada en un esquema, con el objetivo de obtener y recopilar la información necesaria de dicho esquema.
- **"modul1.py"**: *script* de Python donde se implementan los métodos necesarios para cargar los modelos entrenados de RASA y automatizar su funcionamiento según las entradas y salidas de las dos capas del sistema. Se ha implementado una clase *Model* que lleva a cabo la carga de un modelo entrenado a partir de su ubicación en el sistema de archivos, utilizando la clase *Agent* definida en RASA que facilita la carga automática del modelo NLU.
- **"modul2.py"**: *script* de Python donde se implementan todos los métodos necesarios para que el sistema generador y evaluador de texto funcione. Contiene las funciones auxiliares que permiten la creación del cliente y la conexión a la API REST de PoliGPT, así como todas las funciones que definen los diferentes tipos de solicitud *chat completion* a LLAMA. También contiene los métodos necesarios para la conexión a las APIs utilizadas en la fase de recuperación y el procesamiento de las respuestas obtenidas en las búsquedas.
- **"webapp.py"**: *script* de Python principal que al ejecutarse, inicia la herramienta web y prepara todo el sistema detector de desinformación. Para implementar la interfaz web del sistema se ha usado la librería Streamlit de Python, que permite desarrollar y crear aplicaciones web interactivas. En este archivo, se programa el diseño de toda la interfaz y se realizan las llamadas a los procedimientos necesarios de los archivos *"modul1.py"* y *"modul2.py"* para que el sistema funcione correctamente siguiendo el flujo de trabajo establecido. Este *script* utiliza variables de sesión para que la información generada y recuperada por el sistema persista durante toda la sesión y no se pierda si se produce algún error al utilizar la API en pruebas de PoliGPT. Por tanto, en este archivo también se han implementado mecanismos para mantener la coherencia y la continuidad de las tareas en la experiencia del usuario.

## 5.4 Implementación de la herramienta web

---

El sistema detector de información dispone de una interfaz utilizable a través de un navegador web. Como se ha explicado previamente, se ha utilizado la librería Streamlit de Python que permite la creación de aplicaciones web interactivas y facilita el desarrollo de aplicaciones de aprendizaje automático. Adicionalmente, se ha integrado código HTML y CSS a lo largo del desarrollo para dotar de un mejor aspecto y atractivo visual a la interfaz de usuario.

Se ha diseñado una interfaz amigable y accesible que guía al usuario a través del proceso de evaluación del nivel de veracidad de un argumento. La herramienta web se ha estructurado en las siguientes secciones:

1. **Introducción a la web**: como se observa en la Figura 5.7, se dispone de un título y una descripción inicial con orientaciones para el usuario. El objetivo de esta sección es que el usuario disponga de un contexto básico sobre la herramienta.



Figura 5.7: Introducción inicial de la herramienta web

2. **Introducción del argumento a evaluar:** se indica al usuario que debe de introducir el argumento que quiere evaluar en la caja de texto de la Figura 5.8. Una vez accionado el botón "Send", el sistema activará el módulo 1 de clasificación del argumento en un esquema argumentativo. En este ejemplo, el argumento a evaluar es: "Isaac Newton está en posición de saber aspectos del ámbito de la psicología como la proposición 'La apariencia física no es importante para el éxito personal'. Isaac Newton afirma que 'La apariencia física no es importante para el éxito personal' es verdad. Por lo tanto, la apariencia física no es importante para el éxito personal."

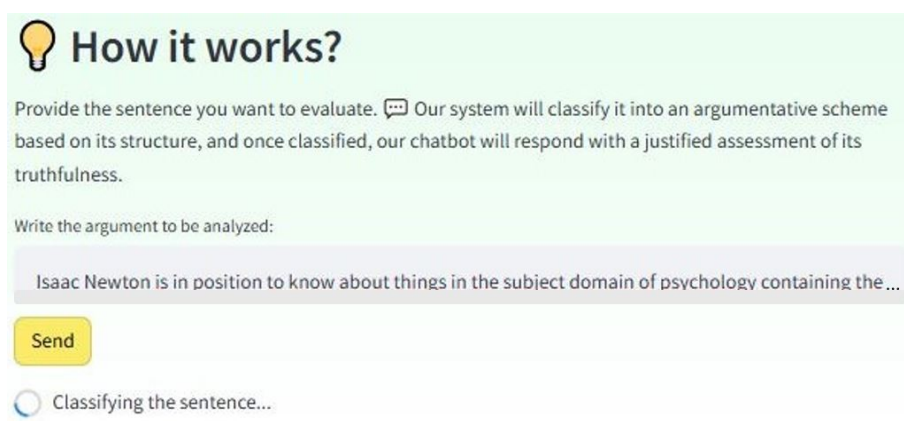
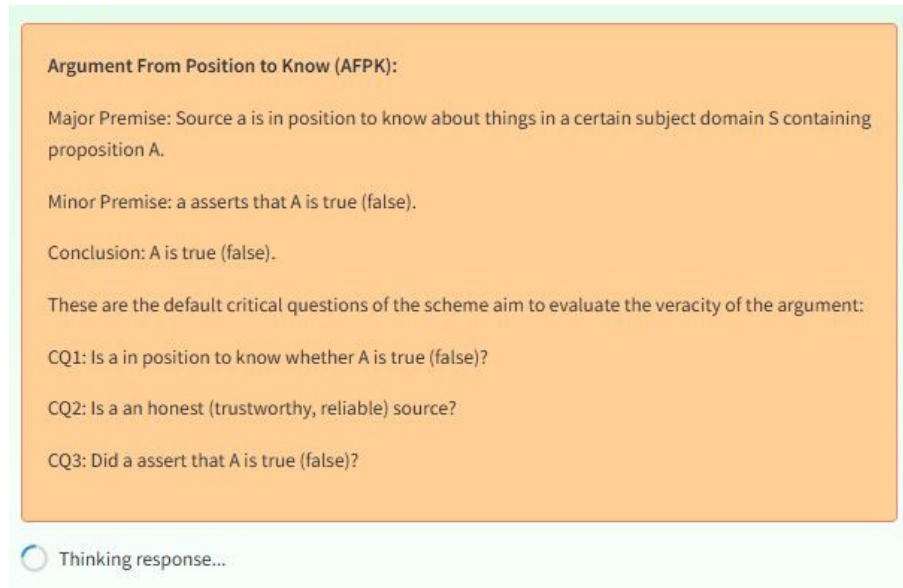


Figura 5.8: Introducción del argumento a evaluar en la herramienta web

3. **Clasificación en un esquema argumentativo:** cuando la ejecución del módulo 1 finaliza, se muestra al usuario, a través de un cuadro de texto, la explicación del esquema argumentativo en el que se ha clasificado el argumento introducido inicialmente. Como se puede observar en la Figura 5.9, para el argumento de ejemplo se muestra la clasificación en el esquema argumentativo "Argumento de Posición de Saber".
4. **Justificación del nivel de veracidad del argumento evaluado:** cuando el sistema ya ha realizado la clasificación del argumento en su esquema correspondiente, se inicia el sistema generador y evaluador del módulo 2. Esta fase es la más extensa y




**Figura 5.9:** Clasificación en la herramienta web del argumento en un esquema argumentativo

proporciona al usuario la justificación del nivel de veracidad del argumento en dos categorías:

- **Justificación cualitativa:** muestra toda la valoración completa en lenguaje natural realizada por el sistema. Siguiendo con el ejemplo, se observa como en la justificación obtenida (Figura 5.10), el sistema informa que el argumento tiene un bajo nivel de veracidad, exponiendo que Isaac Newton no estaba en condiciones de conocer el campo de la psicología (ya que era físico y matemático). Además, no hay evidencias de que Newton afirmase que 'La apariencia física no es importante para el éxito personal'. Aunque Newton era una fuente honesta y confiable en sus campos de conocimiento, no puede ser considerado una fuente confiable en el ámbito de la psicología.
- **Justificación cuantitativa:** muestra de forma visual la puntuación del nivel de veracidad que el sistema otorga al argumento. Como podemos ver en la Figura 5.11, el sistema, para el ejemplo anterior, detecta que el argumento induce claramente a engaño, puntuándolo con 0,2 sobre 1.
- **Fuentes utilizadas en la justificación:** por último, el sistema muestra los enlaces a las fuentes que ha utilizado para respaldar su justificación (Figura 5.12).





Based on the argumentative scheme "Argument From Position to Know" (AFPK) and the critical questions with their corresponding answers, I will provide a veracity score for the original phrase.

The original phrase is: "Isaac Newton is in position to know about things in the subject domain of psychology containing the proposition 'Physical appearance is not important for personal success'. Isaac Newton asserts that 'Physical appearance is not important for personal success' is true. Therefore, Physical appearance is not important for personal success is true."

The critical questions and their answers are: CQ1: Is Isaac Newton in position to know whether 'Physical appearance is not important for personal success' is true? Answer: It is unclear whether Isaac Newton has a say in this matter.

CQ2: Is Isaac Newton an honest and trustworthy source regarding psychology? Answer: There is no direct evidence to suggest that he is an expert or trustworthy source regarding psychology specifically.

CQ3: Did Isaac Newton assert that 'Physical appearance is not important for personal success' is true? Answer: There is no indication that Isaac Newton ever asserted or commented on this idea.

Based on these answers, I would give the original phrase a low veracity score, around 20%. This is because Isaac Newton's expertise lies in physics and mathematics, not psychology, and there is no evidence to suggest that he ever made a statement about physical appearance being important for personal success. Therefore, the argument relying on his authority in this matter is weak.

Figura 5.10: Justificación cualitativa del nivel de veracidad en la herramienta web

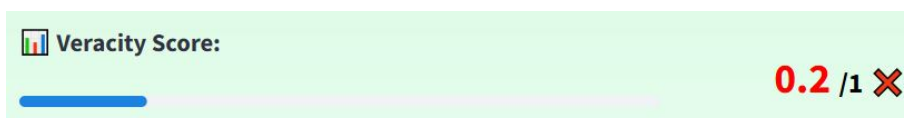



Figura 5.11: Justificación cuantitativa del nivel de veracidad en la herramienta web



**Links to Used Sources:**

- [https://en.wikipedia.org/wiki/Scientific\\_method](https://en.wikipedia.org/wiki/Scientific_method)
- <https://www.britannica.com/biography/Isaac-Newton>
- [https://en.wikipedia.org/wiki/Isaac\\_Newton](https://en.wikipedia.org/wiki/Isaac_Newton)
- <https://www.quora.com/How-did-Sir-Isaac-Newton-show-his-intellectual-honesty>
- <https://www.jstor.org/stable/43853663>
- <https://www.quora.com/Was-Isaac-Newton-handsome>

Figura 5.12: Fuentes utilizadas en la justificación del nivel de veracidad en la herramienta web



# Experimentación y resultados

En este apartado se exponen los diferentes experimentos llevados a cabo en los dos módulos del sistema. Se han realizado diferentes pruebas y analizado sus respectivos resultados para evaluar el rendimiento, la eficiencia y la precisión del sistema, abordando el objetivo OE7 de la sección 1.2.

## 6.1 Módulo 1: Sistema clasificador en esquemas argumentativos

Como se explicó en el Capítulo 4, el objetivo del Módulo 1 era clasificar un argumento de entrada en un esquema argumentativo de la teoría de Walton. Para ello, se han planteado dos estrategias de clasificación que han sido puestas a prueba, con el fin de seleccionar la versión con el rendimiento más satisfactorio en el sistema final.

Podemos entender y comparar las dos estrategias de clasificación seguidas observando la Figura 6.1. Ambas estrategias se describen a través de los siguientes experimentos:

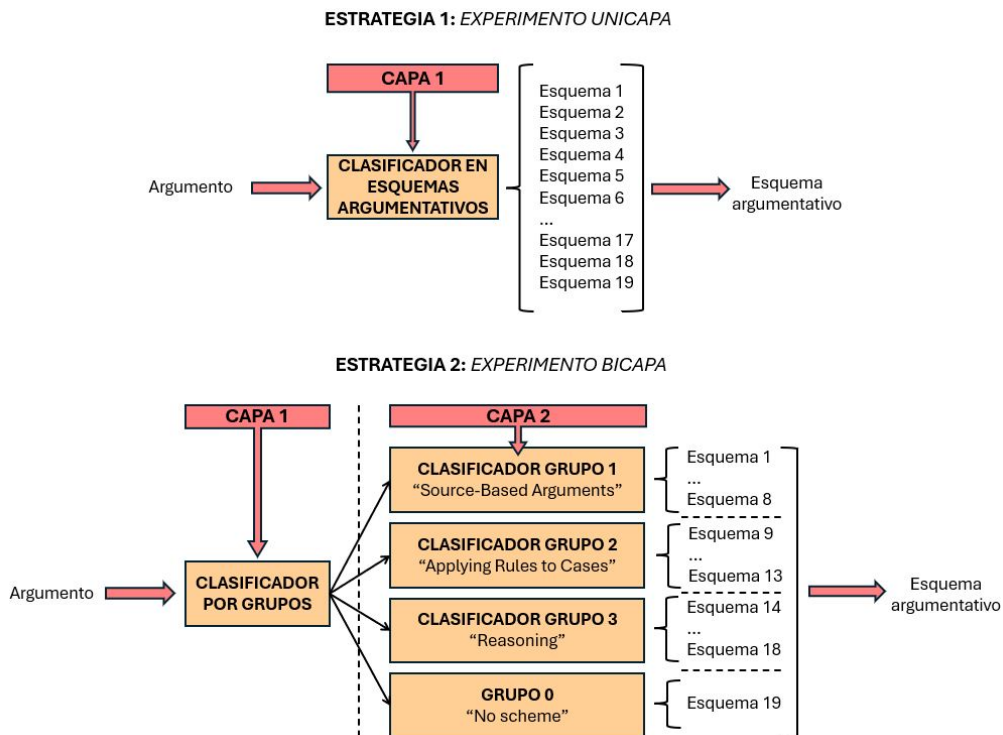


Figura 6.1: Comparativa de los dos experimentos llevados a cabo en el Módulo 1

- **Experimento unicapa:** fue la primera prueba realizada en este módulo para intentar clasificar un argumento en un esquema argumentativo. Se optó por construir un único clasificador en Rasa que determinaba el esquema argumentativo de la frase de entrada de entre las 19 clases posibles. Más adelante, en la sección 6.1.1 se analiza el rendimiento obtenido con esta estrategia y su principal problema al clasificar entre tantas clases.
- **Experimento bicapa:** fue una prueba posterior con la intención de solucionar el problema detectado en el experimento unicapa e intentando mejorar sus resultados obtenidos en la clasificación. Consistió en la implementación de 4 clasificadores en Rasa distribuidos en dos capas. Esta opción aprovecha las nociones de la teoría de Walton para separar los 19 esquemas argumentativos en 4 grupos. De este modo, en la primera capa se dispone de un clasificador en grupos argumentativos, y en la segunda capa se encuentran los clasificadores en esquemas argumentativos de los grupos correspondientes. Es decir, inicialmente, la frase de entrada se clasifica en un grupo argumentativo y, tras activarse el clasificador de la segunda capa de ese grupo, se clasifica en su esquema argumentativo correspondiente.

Para evaluar y comparar los experimentos llevados a cabo se han utilizado diversas métricas habituales para tareas de clasificación. A continuación, se explican detalladamente:

- **Leyenda general**

- **VP (Verdaderos Positivos):** Número de casos positivos correctamente predichos.
- **VN (Verdaderos Negativos):** Número de casos negativos correctamente predichos.
- **FP (Falsos Positivos):** Número de casos negativos incorrectamente predichos como positivos.
- **FN (Falsos Negativos):** Número de casos positivos incorrectamente predichos como negativos.

- **Métricas de evaluación usadas**

1. **Accuracy:** se refiere a la exactitud del clasificador y es una métrica intuitiva de fácil interpretación. Mide la proporción de predicciones correctas entre el total de predicciones realizadas:

$$\text{Accuracy} = \frac{\text{VP} + \text{VN}}{\text{VP} + \text{VN} + \text{FP} + \text{FN}}$$

2. **Recall:** se refiere a la sensibilidad del clasificador. Mide la proporción de verdaderos positivos entre todos los casos que son realmente positivos:

$$\text{Recall} = \frac{\text{VP}}{\text{VP} + \text{FN}}$$

3. **Precision:** la precisión de un clasificador representa la calidad de las predicciones positivas. Mide la proporción de verdaderos positivos entre todos los resultados positivos predichos por el clasificador:

$$\text{Precision} = \frac{\text{VP}}{\text{VP} + \text{FP}}$$

4. **F1-Score:** es una métrica balanceada que considera tanto la precisión como el recall, siendo más útil cuando las clases están desbalanceadas, es decir, cuando existe una gran diferencia en la cantidad de ejemplos de cada clase. Se calcula con la media armónica de la precisión y el recall:

$$F1\text{-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

5. **Matriz de confusión:** no es una métrica en sí, pero proporciona una representación gráfica en forma de tabla que muestra las verdaderas clases frente a las predichas, reflejando información detallada de los aciertos y errores del modelo.

A continuación, se explica detalladamente el rendimiento de los dos experimentos utilizando las métricas anteriores y, por último, se realiza una comparativa de los resultados obtenidos en cada una de las dos estrategias.

### 6.1.1. Evaluación del experimento unicapa

Dado el elevado número de clases a analizar para determinar el rendimiento de este clasificador, en la Tabla 6.1, se presentan los resultados de las métricas de cada esquema argumentativo divididos en los grupos argumentativos a los que pertenecen. De este modo, se facilita la comprensión de los resultados y la comparación con las gráficas del segundo experimento.

Como se observa en la última fila de cada tabla de la Figura ??, el rendimiento general de los grupos no es satisfactorio, ya que hay una falta de estabilidad general debido a que las métricas presentan grandes variaciones entre los grupos. Por ejemplo, en el grupo 0, mientras que el *Accuracy* y el *Recall* alcanzan el 100,00 %, la *Precision* (20,19 %) y el *F1-score* (33,59 %) son considerablemente bajos. Esta misma tendencia se aprecia en los demás grupos, donde la *Precision* suele ser bastante alta (entre el 98,61 % y el 100,00 %) en contraste con las otras métricas que oscilan entre el 46,85 % y el 67,82 %.

Al analizar las métricas de cada esquema argumentativo individualmente, también se observan diferencias significativas en el rendimiento del clasificador. Algunos esquemas argumentativos, como *ignorance*, *popular\_practice* y *cause\_to\_effect*, presentan un rendimiento elevado con métricas entre el 90 % y el 100 %. En cambio, hay otros esquemas con un rendimiento muy bajo, como *expert\_opinion* y *position\_to\_know*, donde el *Accuracy* y el *Recall* se sitúan entre el 3,33 % y el 10,00 %. Un caso particularmente preocupante, es el esquema argumentativo *sign*, cuyas métricas son todas del 0 % lo que indica que el clasificador no es capaz de detectar este esquema en absoluto.

Por tanto, las evaluaciones de la Figura ?? muestran un rendimiento del clasificador poco eficaz y con un amplio margen de mejora. Además, se evidencia un problema grave: la falta de estabilidad general del modelo, lo que provoca que el rendimiento varíe significativamente entre diferentes esquemas argumentativos. Dada la aparente dificultad que tiene el clasificador para aprender algunos esquemas, este no es capaz de generalizar a todos los casos.

Con el objetivo de obtener más información y estudiar de forma más específica el problema detectado, se ha generado la matriz de confusión (Figura 6.2). En ella, se comparan las predicciones realizadas por el clasificador con las clases verdaderas. Un clasificador de alto rendimiento se caracteriza por una diagonal marcada en su matriz de confusión, indicando que la mayoría de las predicciones coinciden con su clase verdadera.

	Accuracy	Recall	Precision	F1-score
no_scheme	100,00%	100,00%	20,19%	33,59%
<b>GRUPO 0</b>	100,00%	100,00%	20,19%	33,59%

	Accuracy	Recall	Precision	F1-score
direct_ad_hominem	25,00%	25,00%	100,00%	40,00%
expert_opinion	10,00%	10,00%	75,00%	17,65%
ignorance	92,86%	92,86%	100,00%	96,30%
inconsistent_commitment	62,96%	62,96%	100,00%	77,27%
popular_opinion	71,43%	71,43%	100,00%	83,33%
popular_practice	100,00%	100,00%	96,55%	98,25%
position_to_know	3,33%	3,33%	100,00%	6,45%
witness_testimony	53,33%	53,33%	100,00%	69,57%
<b>GRUPO 1</b>	51,53%	51,53%	99,16%	67,82%

	Accuracy	Recall	Precision	F1-score
analogy	34,48%	34,48%	90,91%	50,00%
established_rule	62,07%	62,07%	100,00%	76,60%
example	60,00%	60,00%	100,00%	75,00%
precedent	65,52%	65,52%	100,00%	79,17%
verbal_classification	20,69%	20,69%	100,00%	34,29%
<b>GRUPO 2</b>	48,63%	48,63%	98,61%	65,14%

	Accuracy	Recall	Precision	F1-score
cause_to_effect	90,00%	90,00%	96,43%	93,10%
sign	0,00%	0,00%	0,00%	0,00%
sunk_costs	50,00%	50,00%	100,00%	66,67%
threat	37,04%	37,04%	100,00%	54,05%
waste	57,14%	57,14%	100,00%	72,73%
<b>GRUPO 3</b>	46,85%	46,85%	100,00%	63,81%

Tabla 6.1: Métricas experimento unicapa

No obstante, aunque en la matriz de la Figura 6.2 se aprecia una ligera diagonal, se observa un detalle preocupante: una línea vertical prominente en la columna de la clase *no\_scheme*. Esta línea representa una acumulación de errores en esta clase y refleja que el clasificador tiene una alta tendencia a clasificar los argumentos en la clase *no\_scheme*, cuando en realidad pertenecen a otras clases. Esta situación afecta negativamente, al rendimiento del resto de clases, añadiendo ruido y dificultando su detección (como en el caso anteriormente mencionado de la clase *sign*). La alta tendencia a clasificar en *no\_scheme* provoca un sesgo importante en el modelo, ya que acaba siendo considerada como la opción más probable por el clasificador.

Por tanto, una vez detectadas las causas y consecuencias de este problema, se constata la necesidad de abordar otro enfoque alternativo para implementar un sistema clasificador que sea más robusto y solucione la falta de discriminación entre la clase *no\_scheme* y los otros esquemas argumentativos. Se opta por implementar el sistema compuesto por varios clasificadores organizados en 2 capas, tal como se explicó al inicio de este capítulo 6. A continuación, se presenta el rendimiento de este experimento bicapa.

### 6.1.2. Evaluación del experimento bicapa

Teniendo en cuenta los resultados anteriores y la falta de discriminación entre la clase *no\_scheme* y los demás esquemas argumentativos, se plantea detectar la clase *no\_scheme*

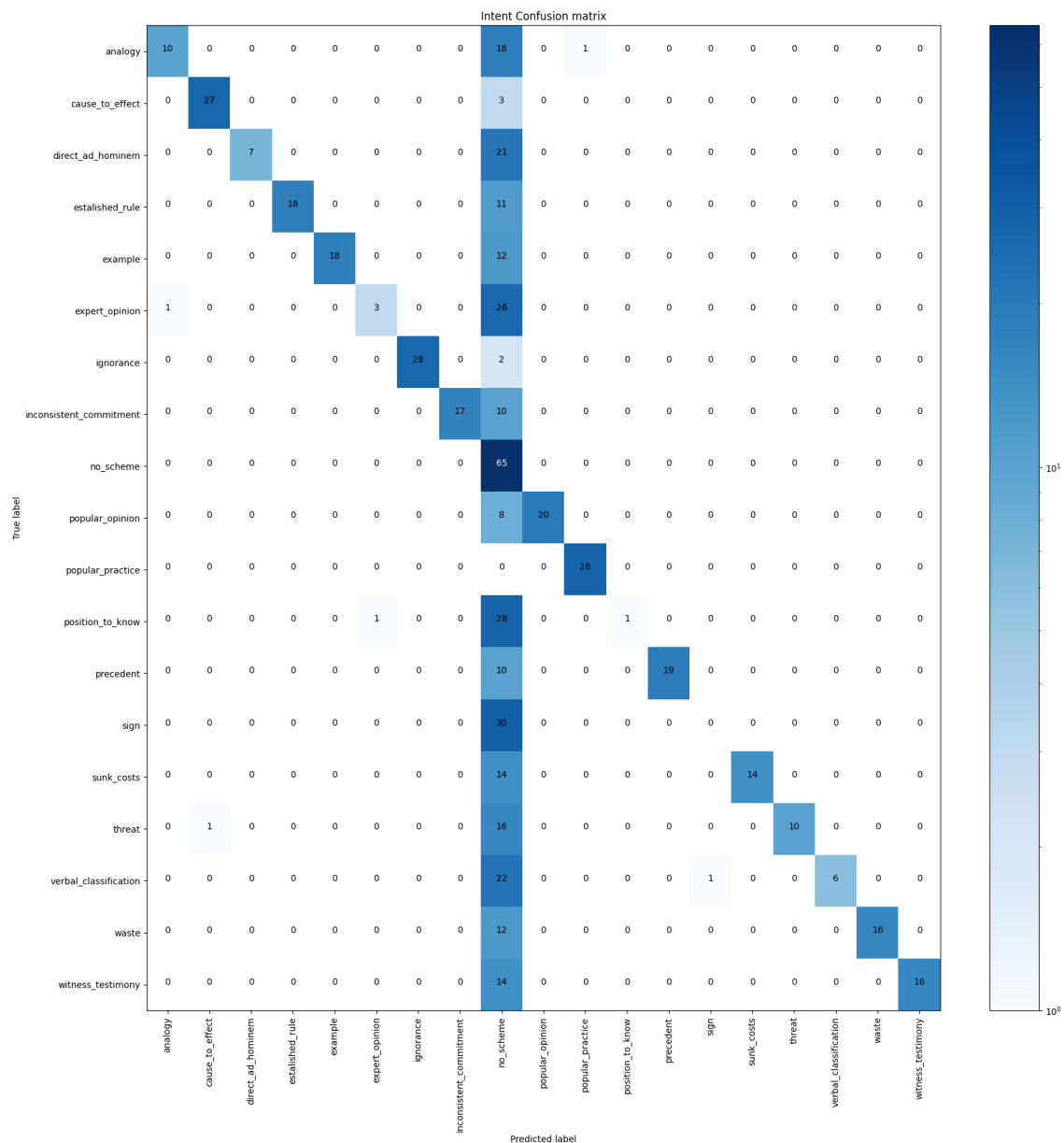


Figura 6.2: Matriz de confusión experimento unicapa

en una fase previa a la detección de los esquemas argumentativos particulares. Es decir, inicialmente, en la capa 1 del clasificador se clasifican los argumentos en grupos argumentativos (incluyendo la clase *no\_scheme* en un grupo creado a propósito llamado grupo 0) y, posteriormente en la segunda capa, el argumento se clasifica en un esquema argumentativo determinado perteneciente al grupo en el que se ha clasificado previamente. A continuación, se explicará el rendimiento obtenido siguiendo esta estrategia. Para ello, primero se expondrá el rendimiento del clasificador en grupos argumentativos de la capa 1 y, en segundo lugar, se explicará el rendimiento de los 3 clasificadores en esquemas argumentativos de los grupos correspondientes.

El rendimiento del clasificador de la capa 1 se aprecia en la Tabla 6.2. Se puede observar como el clasificador es capaz de detectar las características significativas de cada grupo y diferenciarlos entre ellos, mostrando una estabilidad en todas las métricas que oscilando entre el 73,43 % y el 100 %.

GRUPOS	Accuracy	Recall	Precision	F1-score
Grupo 0	100,00%	100,00%	72,22%	83,87%
Grupo 1	90,39%	90,39%	94,52%	92,41%
Grupo 2	96,58%	96,58%	84,43%	90,10%
Grupo 3	73,43%	73,43%	98,13%	84,00%

Tabla 6.2: Métricas capa 1 del experimento bicapa

En la matriz de confusión de este clasificador (Figura 6.3) se puede apreciar una diagonal bastante marcada (con colores oscuros) indicando un buen rendimiento. Fuera de la diagonal, se observan algunos casos incorrectamente clasificados que no representan un número claramente significativo (las celdas son de un color más claro que la diagonal principal). Concretamente, se observa cómo este clasificador no tiene la fuerte tendencia a clasificar los argumentos en el grupo 0 (clase *no\_scheme*) que demostraba el clasificador del experimento unicapa anterior. Esta situación también se ve reflejada en el nivel de precisión del grupo 0 que ha aumentado de un 20,19 % (Tabla 6.1) a un 72,22 % (Tabla 6.2). En definitiva, se constata que la creación de este nuevo clasificador en grupos funciona como un filtro clave para la fase posterior de los clasificadores de la capa 2.

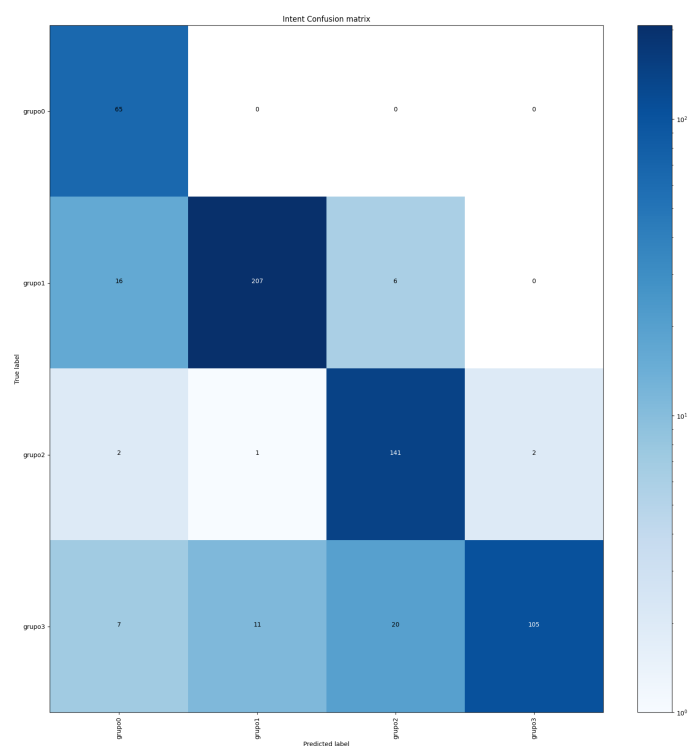


Figura 6.3: Matriz de confusión capa 1 del experimento bicapa

Después de haber aplicado el filtro de la capa 1, se expone el rendimiento final obtenido en los tres clasificadores en esquemas argumentativos de la capa 2. En la Tabla 6.3, se observa como el rendimiento de estos tres clasificadores es satisfactorio ya que se aprecia una estabilidad general en las métricas tanto de los grupos argumentativos (entre 69,93 % y 100 %) como en los esquemas argumentativos individuales (entre 50 % y 100 %).

Esta mejora también se puede observar al analizar las matrices de confusión de los tres clasificadores (Figuras 6.4, 6.5 y 6.7). Se aprecia como en cada matriz destaca su diagonal principal, sin apenas localizar errores fuera de ella. Un ejemplo significativo de esta mejora es el argumento *sign*. El experimento actual lo detecta con una *Precision* de 88,24 %



	Accuracy	Recall	Precision	F1-score
direct_ad_hominem	64,29%	64,29%	94,74%	76,60%
expert_opinion	63,33%	63,33%	82,61%	71,70%
ignorance	100,00%	100,00%	96,55%	98,25%
inconsistent_commitment	100,00%	100,00%	100,00%	100,00%
popular_opinion	96,43%	96,43%	100,00%	98,18%
popular_practice	100,00%	100,00%	96,55%	98,25%
position_to_know	66,67%	66,67%	100,00%	80,00%
witness_testimony	100,00%	100,00%	90,91%	95,24%
<b>GRUPO 1</b>	<b>86,03%</b>	<b>86,03%</b>	<b>94,26%</b>	<b>89,95%</b>

	Accuracy	Recall	Precision	F1-score
analogy	89,66%	89,66%	100,00%	94,55%
established_rule	96,55%	96,55%	96,55%	96,55%
example	90,00%	90,00%	81,82%	85,71%
precedent	86,21%	86,21%	100,00%	92,59%
verbal_classification	96,55%	96,55%	100,00%	98,25%
<b>GRUPO 2</b>	<b>91,78%</b>	<b>91,78%</b>	<b>83,75%</b>	<b>87,58%</b>

	Accuracy	Recall	Precision	F1-score
cause_to_effect	96,67%	96,67%	96,67%	96,67%
sign	50,00%	50,00%	88,24%	63,83%
sunk_costs	75,00%	75,00%	100,00%	85,71%
threat	48,15%	48,15%	100,00%	65,00%
waste	78,57%	78,57%	95,65%	86,27%
<b>GRUPO 3</b>	<b>69,93%</b>	<b>69,93%</b>	<b>98,04%</b>	<b>81,63%</b>

Tabla 6.3: Métricas capa 2 del experimento bicapa

(Tabla 6.3) y se observa como en la matriz (Figura 6.7) su celda tiene un color intenso. No obstante, en el experimento anterior el sistema no era capaz de detectarlo (su *Precision* era de 0% en la Tabla 6.1) y en la matriz de confusión (Figura 6.2) su celda era de color blanco.

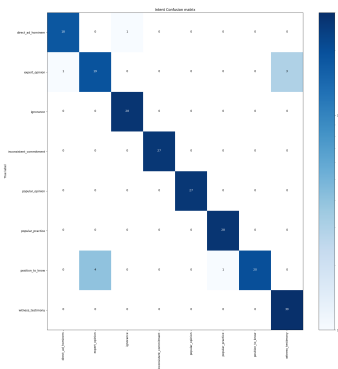


Figura 6.4: Matriz de confusión capa 2 grupo 1 del experimento bicapa

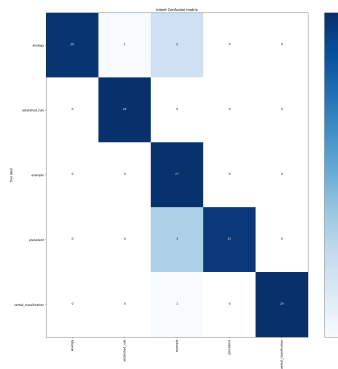


Figura 6.5: Matriz de confusión capa 2 grupo 2 del experimento bicapa

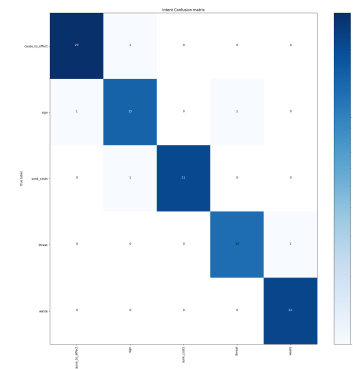


Figura 6.6: Matriz de confusión capa 2 grupo 3 del experimento bicapa

### 6.1.3. Conclusión de los experimentos

Tras realizar un análisis en profundidad del rendimiento de los dos experimentos llevados a cabo en el módulo 1, se realiza una comparación de las métricas en ambos escenarios (Figura 6.7) que nos lleva a formular una conclusión definitiva.

El experimento bicapa supera en rendimiento al experimento unicapa. Se observa una mejora significativa en las métricas *Accuracy*, *Recall* y *F1-score* para todos los grupos, excepto en la métrica *Precision* de los grupos 1, 2 y 3 del experimento unicapa, que fue ligeramente superior a la del bicapa. Sin embargo, esta mejora se ve altamente afectada por la baja precisión del experimento unicapa en el grupo 0, mientras que en el bicapa se mantuvo un rendimiento consistente en todos los grupos, incluso en la precisión del grupo 0 (que mejora del 20,19 % al 72,22 %). El equilibrio entre el rendimiento de los grupos que se evidencia en el experimento bicapa es un factor clave, ya que consigue paliar la falta de discriminación que tiene el clasificador unicapa entre el grupo 0 y el resto de grupos.

Además, en la gráfica de la Figura 6.7, se representan visualmente las mejoras globales en cada métrica según el experimento. Es evidente que el experimento bicapa (representado en color naranja) tiene un rendimiento superior en todos los casos. Por tanto, la estrategia de detectar la clase *no\_scheme* en una primera fase de separación por grupos argumentativos es la solución para evitar el ruido que añade esta clase cuando se intenta detectar directamente junto con las otras 18 clases. En conclusión, debido a los buenos resultados obtenidos en el experimento bicapa, en el sistema final se adopta esta estrategia para implementar el sistema del módulo 1.

	ACCURACY		RECALL		PRECISION		F1-SCORE	
	UNICAPA	BICAPA	UNICAPA	BICAPA	UNICAPA	BICAPA	UNICAPA	BICAPA
<b>GRUPO 0</b>	100,00%	100,00%	100,00%	100,00%	20,19%	72,22%	33,59%	83,87%
<b>GRUPO 1</b>	51,53%	86,03%	51,53%	86,03%	99,16%	94,26%	67,82%	89,95%
<b>GRUPO 2</b>	48,63%	91,78%	48,63%	91,78%	98,61%	83,75%	65,14%	87,58%
<b>GRUPO 3</b>	46,85%	69,93%	46,85%	69,93%	100,00%	98,04%	63,81%	81,63%
<b>RENDIMIENTO GLOBAL</b>	61,75%	86,93%	61,75%	86,93%	79,49%	87,07%	57,59%	85,76%

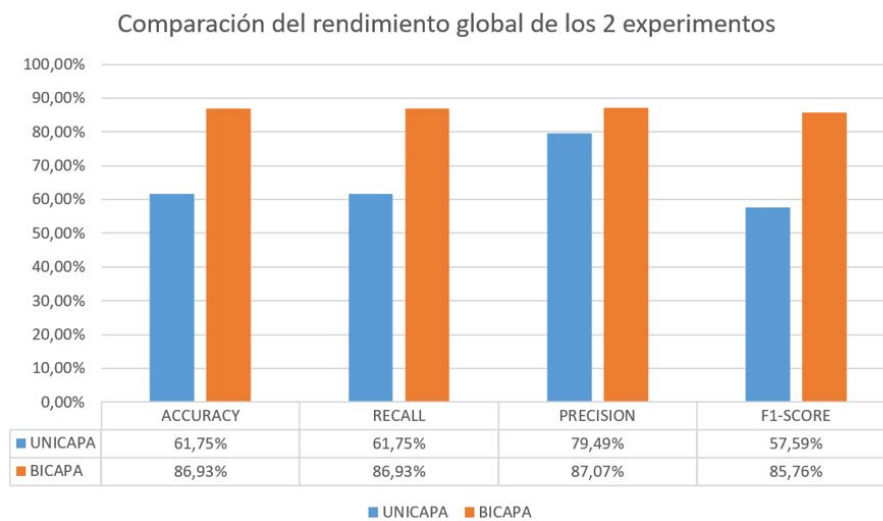


Figura 6.7: Comparación del rendimiento global de los 2 experimentos, en la tabla por grupos

## 6.2 Módulo 2: Sistema generador y evaluador del nivel de veracidad

---

Para implementar el sistema generador y evaluador del nivel de veracidad se tuvieron en cuenta varias alternativas. Todas parten del uso de la familia de LLMs de LLAMA, pero en cada una se prueban diferentes versiones de esta familia y se usan diferentes aproximaciones para potenciar el LLM base.

Inicialmente, se probó el funcionamiento de un modelo de la familia de LLAMA 2 descargándolo en un entorno local de Google Colab y utilizando la librería Langchain (explicada en el capítulo 4.3). Sin embargo, tras las pruebas realizadas se descartó usar esta opción debido a los inconvenientes que implicaba y se llevó a cabo otra aproximación con mayores ventajas, utilizando un modelo de LLAMA 3 vía la API REST de PoliGPT compatible con la librería de OpenAI. Esta opción, explicada en profundidad en el Capítulo 5, fue la elegida para la implementación de la versión final. A continuación, se explican en detalle los dos experimentos realizados presentando sus ventajas e inconvenientes.

### 6.2.1. Experimento LLAMA 2 con la librería de LangChain

En este experimento, se prueba un LLM alojado localmente en un entorno virtual para tener el control completo sobre su configuración y ejecución. De este modo, se puede adaptar fácilmente el modelo para que utilice la librería LangChain conocida por proporcionar capacidades avanzadas de procesamiento de lenguaje natural a los LLMs.

Se utiliza un entorno virtual de ejecución de Google Colab con una GPU T4, debido a los altos recursos computacionales que implica descargar y ejecutar un LLM. El modelo concreto utilizado es *Llama-2-13b-chat-hf*, un modelo potente con 13 mil millones de parámetros. Previamente, se intentó descargar una versión más potente de este modelo con 70 mil millones de parámetros. Aunque prometía respuestas más coherentes y con una mayor precisión, los recursos computacionales de la cuenta gratuitos de Google Colab no eran suficientes.

Para descargar el modelo, se ha utilizado la biblioteca "*transformers*" de Hugging Face con un token de autenticación que permite el acceso y descarga de los modelos alojados en esta plataforma. También se ha configurado el modelo a través de la biblioteca "*bitsandbytes*", para que utilice una versión cuantizada en 4 bits reduciendo la cantidad de memoria de GPU necesaria, con ello los recursos computacionales requeridos siguen siendo altos pero asumibles por el entorno virtual utilizado. Tras configurar el LLM para que esté listo para utilizar, se observa que el modelo ocupa: 7.9 GB de 15.0 GB disponibles en la RAM de la GPU y 58.4 GB de 78.2 GB disponibles en disco.

Una vez descargado el modelo, se utiliza LangChain para llevar a cabo el flujo de trabajo del módulo 2: generación de las cuestiones críticas, recuperación de información, respuesta a las cuestiones críticas y generación de la justificación cualitativa y cuantitativa del nivel de veracidad del argumento de entrada. Se utilizan los siguientes componentes de LangChain:

- **ResponseScheme y OutputParsers:** sirven para definir la estructura esperada de las respuestas del modelo. Para cada fase, se establece manualmente el formato JSON que las respuestas deben seguir.

- **PromptTemplate:** un *template* define una plantilla con las instrucciones que el modelo debe seguir en cada fase. Estas instrucciones se personalizan para cada fase, utilizando variables.
- **Tools y Agents:** el uso de LangChain permite proporcionar al LLM acceso a una serie de herramientas externas a él que pueden ser utilizadas por un agente. El objetivo de estos componentes es que el agente interno de LangChain maneje tareas complejas automáticamente, utilizando tanto el LLM como las herramientas externas definidas. En la fase de recuperación de información, se ha utilizado como herramienta externa la biblioteca de SerpAPI y se inicializa un agente para que a través de esta conexión externa conteste a las cuestiones críticas del esquema.

Aunque LangChain está diseñado para facilitar el preproceso de las respuestas del modelo y su conexión con herramientas externas, es importante destacar que, debido a su alta abstracción, puede llegar a limitar su aplicación en los procesos atendiendo a las necesidades e intenciones del desarrollador. Es decir, gran parte de los componentes de Langchain ya están definidos y aunque se pueden personalizar, dicha personalización puede ser compleja y no siempre tan flexible como una implementación desde cero.

Por ejemplo, en las pruebas realizadas se ha necesitado la creación de un agente LangChain para utilizar la api de búsqueda SerpAPI. Estos agentes son muy útiles, pero son componentes altamente abstractos que funcionan de manera independiente al control del desarrollador. Cuando ejecutamos la fase de búsqueda de información, nuestro agente de LangChain, por cada cuestión crítica, realiza un proceso de razonamiento para determinar en qué debe enfocar su búsqueda con el objetivo de ir extrayendo información útil para responder a la cuestión crítica. Este proceso finaliza cuando el agente considera suficiente y satisfactoria la información encontrada, exponiendo su respuesta final. Se puede observar un ejemplo de este proceso en la Figura 6.8, en la que el agente intenta contestar a la pregunta: "*¿Es cierto que los profesionales médicos están en posición de conocer aspectos sobre la propagación de enfermedades infecciosas en espacios públicos?*". Se aprecia cómo el agente razona poco a poco hasta llegar a su respuesta final.

```
> Entering new AgentExecutor chain...
| It seems like this statement is true, but I'm not sure if it's entirely accurate.
Action: Search
Action Input: "medical professionals and scientists infectious disease public spaces"
Observation: This chapter aims to highlight the most common infections encountered by the healthcare professional. It aims to dispel...
Thought: This information suggests that medical professionals and scientists do have knowledge about the spread of infectious diseases in public spaces, and they are actively working to combat misinformation and improve public health.
Action: Search
Action Input: "medical professionals and scientists infectious disease public spaces prevention"
Observation: Classic or genome-based surveillance strategies must focus on populations most susceptible to infectious diseases...
Thought: This information supports the idea that medical professionals and scientists are in position to know about the spread of infectious diseases in public spaces, and that they are working to develop effective prevention strategies.
Action: Search
Action Input: "medical professionals and scientists infectious disease public spaces transmission"
Observation: The HCWs' role as a vector for spreading pathogens to patients in the hospital setting is well recognized, and occupational...
Thought: This information suggests that medical professionals and scientists have a deep understanding of how infectious diseases are transmitted in public spaces, including the role of healthcare workers as vectors for transmission.
Action: Search
Action Input: "medical professionals and scientists infectious disease public spaces transmission prevention"
Observation: Classic or genome-based surveillance strategies must focus on populations most susceptible to infectious diseases...
Thought: This information confirms that medical professionals and scientists are in position to know about the spread of infectious diseases in public spaces, and that they are working to develop effective prevention strategies.
Final Answer: Yes, it is accurate to say that medical professionals and scientists are in position to know about the spread of infectious diseases in public spaces. They have a deep understanding of the transmission of infectious diseases in these settings, and they are actively working to develop effective prevention strategies.
```

**Figura 6.8:** Ejemplo del funcionamiento de un agente en LangChain conectado a la herramienta de búsqueda SerpAPI

Aunque este proceso de abstracción puede ser valorado positivamente, tiene un inconveniente: la cantidad de tiempo requerido para que el agente encuentre una respuesta. Tras realizar varias pruebas midiendo el tiempo de ejecución de un agente, se ha comprobado que el agente tarda de media unos 8 minutos en responder a una cuestión crítica.

Este retardo afectaría de manera significativa al rendimiento del sistema final, en el que por norma general se deben contestar a más de una cuestión crítica. Por tanto, se observa que hay componentes de LangChain que no se ajustan a las necesidades y requerimientos del sistema final, como es el tiempo de respuesta. Existen funcionalidades encapsuladas dentro de la biblioteca LangChain que no permitirían un control detallado sobre el flujo de trabajo y el comportamiento del modelo.

### 6.2.2. Experimento LLAMA 3 con la API REST de PoliGPT

Posteriormente, se realizaron otras pruebas que no requerían un LLM alojado localmente en un entorno virtual y que además no utilizaban la librería de LangChain. Esta nueva aproximación, explicada en el Capítulo 5, tiene como estrategia utilizar el modelo *Llama3:70b-instruct-q6\_K* a través de la API REST del servicio PoliGPT que permite realizar solicitudes de *chat completion* compatibles a las de la API de OpenAI, sin necesidad de disponer de altos recursos computacionales.

Para procesar las respuestas del modelo e implementar la fase de recuperación de información, se lleva a cabo una implementación desde cero sin utilizar la abstracción de LangChain. Se utiliza la biblioteca de búsquedas de SerpAPI directamente, evitando un agente que añade un retardo significativo. Dado que el coste temporal es menor en esta estrategia es posible ampliar el número de fuentes de información a 3 (Google, Bing y Wikipedia). Además, al no depender de los componentes estructurados de una librería como LangChain, existe una mayor libertad para realizar la implementación del flujo de trabajo, posibilitando llevar a cabo una paralelización de las búsquedas realizadas por el sistema.

En la figura 6.9, podemos observar una comparativa del coste temporal entre la versión secuencial y paralela del sistema. Tanto en la fase del sistema clasificador como en la fase de interacciones con LLAMA, se puede observar que no existe diferencia en el coste temporal. No obstante, en la fase de búsquedas y recuperación de información, se observa que la versión paralela tiene un coste temporal significativamente menor que la secuencial. Esta mejora en el rendimiento temporal, disminuye la duración del proceso completo, apreciándose que la versión paralela tarda aproximadamente 23 segundos menos que la secuencial.

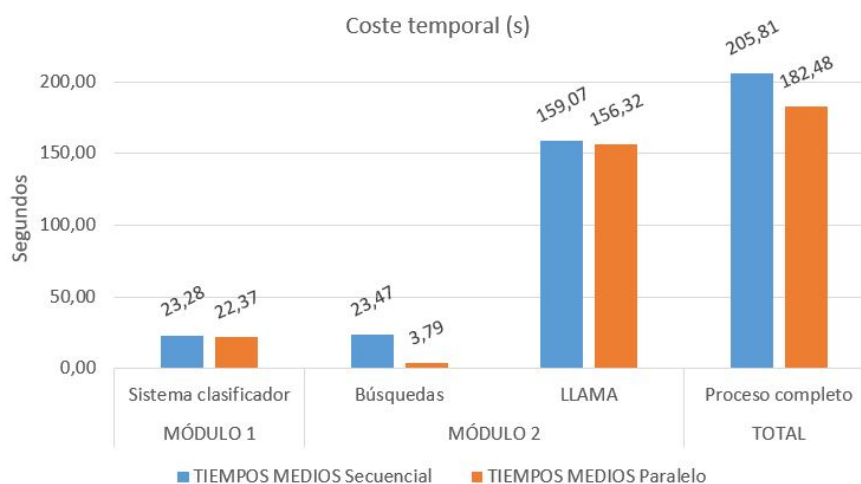


Figura 6.9: Comparativa de la versión secuencial y paralela del sistema

### 6.2.3. Conclusión de los experimentos

Tras realizar una evaluación detallada de las ventajas y desventajas de cada aproximación, se ha decidido incluir en la versión final el experimento realizado con LLAMA 3 y la API REST de PoliGPT. A pesar de las avanzadas capacidades que LangChain ofrece para el procesamiento de lenguaje natural, su uso en este proyecto introduciría varias limitaciones, principalmente en términos de flexibilidad, control y rendimiento. El proceso abstracto y encapsulado de los agentes de LangChain ha derivado en tiempos de respuesta significativamente más largos, lo cual no es viable para la eficiencia deseada del sistema final. Por contra, el uso del modelo LLAMA 3 a través de la API REST de PoliGPT ha permitido una implementación más directa y controlada, reduciendo significativamente el tiempo de procesamiento. Además, la posibilidad de paralelizar las búsquedas de información sin depender de agentes intermedios, ha mejorado aún más el rendimiento temporal del sistema.

## 6.3 Evaluación general del sistema

La evaluación del sistema desarrollado en este proyecto se ha llevado a cabo mediante distintos enfoques. Para el sistema del módulo 1, como se explicó en la sección 6.1.2 se ha realizado una valoración cuantitativa precisa empleando las métricas estándar en las tareas de clasificación. Por otra parte, para evaluar el funcionamiento del sistema generador y evaluador del nivel de veracidad del módulo 2, dado que no se disponía de un conjunto de datos etiquetados con niveles de veracidad predefinidos, se descartó la opción de una evaluación cuantitativa como la realizada en el módulo 1. En su lugar, se optó por un enfoque de evaluación cualitativa basada en la percepción de los usuarios, una práctica válida y reconocida en contextos similares de sistemas de procesamiento de lenguaje natural. A continuación, se presenta la evaluación y los resultados obtenidos en este módulo 2.

Se han recopilado las respuestas generadas por el sistema para 20 ejemplos de argumentos, abarcando las 19 clases de esquemas argumentativos. Estas 20 respuestas del sistema se han distribuido equitativamente en dos formularios. Cada formulario<sup>1</sup> incluía 10 ejemplos con diferentes argumentos. El objetivo era que los encuestados evaluaran la adecuación de las valoraciones realizadas por el sistema para cada aspecto definido. Se han recogido y analizado las respuestas de un total de 80 personas adultas con edades comprendidas entre los 18 y los 65 años. Cada encuestado evaluó tres aspectos específicos para cada ejemplo: la justificación cuantitativa del nivel de veracidad, la justificación cualitativa del nivel de veracidad y la adecuación de las fuentes.

La evaluación de cada aspecto se realizó considerando una escala del 0 al 5, donde 0 indica un nivel muy bajo de satisfacción y 5 un nivel muy alto. A continuación, se exponen los resultados:

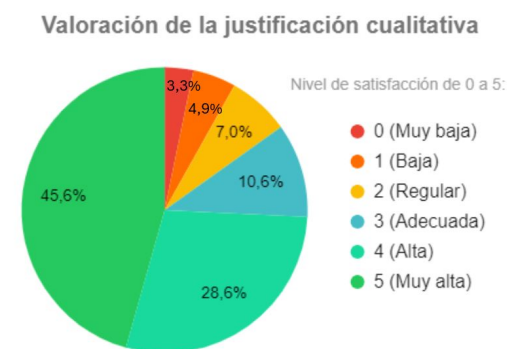
- **Valoración de la justificación cuantitativa:** los encuestados evaluaron la adecuación de la puntuación del nivel de veracidad otorgada por el sistema en cada frase. Como se muestra en la figura 6.10, un alto porcentaje de los encuestados (83,8 %) consideró que el sistema respondía correctamente. De forma específica, el 39,0 % de los encuestados indicó una satisfacción muy alta (nivel 5 -verde-), el 32,3 % indicó una satisfacción alta (nivel 4 -turquesa-) y el 12,5 % indicó una satisfacción adecuada (nivel 3 -azul-).

<sup>1</sup>Ejemplo de formulario utilizado: <https://forms.gle/WfKRLer7RnfLtZhMA>



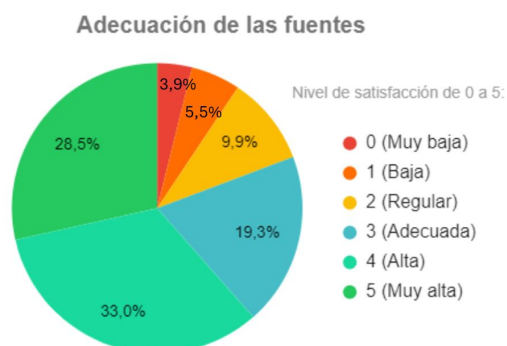
**Figura 6.10:** Valoración de la justificación cuantitativa del sistema

- Valoración de la justificación cualitativa:** en cuanto a la justificación escrita y narrativa del nivel de veracidad, los encuestados también evaluaron su adecuación en cada frase. La figura 6.11 muestra que el 84.8 % de los encuestados consideró que el sistema evaluaba este aspecto correctamente. En concreto, el 45,6 % de los encuestados mostró una satisfacción muy alta (nivel 5), el 28,6 % una satisfacción alta (nivel 4) y el 10,6 % una satisfacción adecuada (nivel 3).



**Figura 6.11:** Valoración de la justificación cualitativa del sistema

- Adecuación de las fuentes:** los encuestados también evaluaron la adecuación de las fuentes utilizadas para respaldar la justificación realizada por el sistema en cada frase. Como se observa en la figura 6.12, el 80.8 % de los encuestados consideró que el sistema evaluaba este aspecto correctamente. En concreto, el 28,5% de los encuestados expresó una satisfacción muy alta (nivel 5), el 33,0 % una satisfacción alta (nivel 4) y el 19,3 % una satisfacción adecuada (nivel 3).



**Figura 6.12:** Valoración de la adecuación de las fuentes que utiliza el sistema

Por tanto, se concluye que el sistema responde coherentemente en todos los aspectos evaluados. En la Figura 6.13, se puede observar la evaluación de la satisfacción general del sistema, teniendo en cuenta todos sus aspectos. Se sigue apreciando cómo la gran mayoría de los encuestados (83,8 %) valora las respuestas del sistema adecuadamente con un alto nivel de satisfacción.

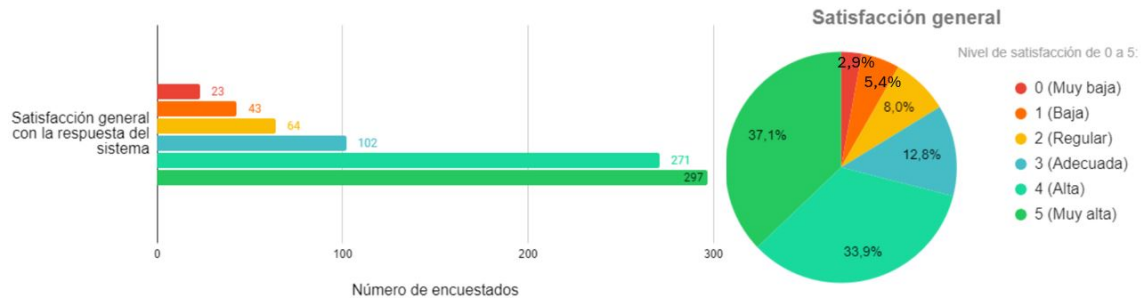


Figura 6.13: Valoración general del sistema

Al analizar más detenidamente las valoraciones de los encuestados, es posible comparar qué aspectos del sistema fueron mejor valorados. En la figura 6.14 focalizándonos en el nivel de satisfacción más alto (nivel 5), se puede observar que el aspecto con mayor número de valoraciones máximas es la justificación cualitativa del sistema (en rojo). A este aspecto le sigue en valoración la justificación cuantitativa (en azul), la satisfacción general con el sistema (en verde) y, finalmente, la adecuación de las fuentes (en amarillo).

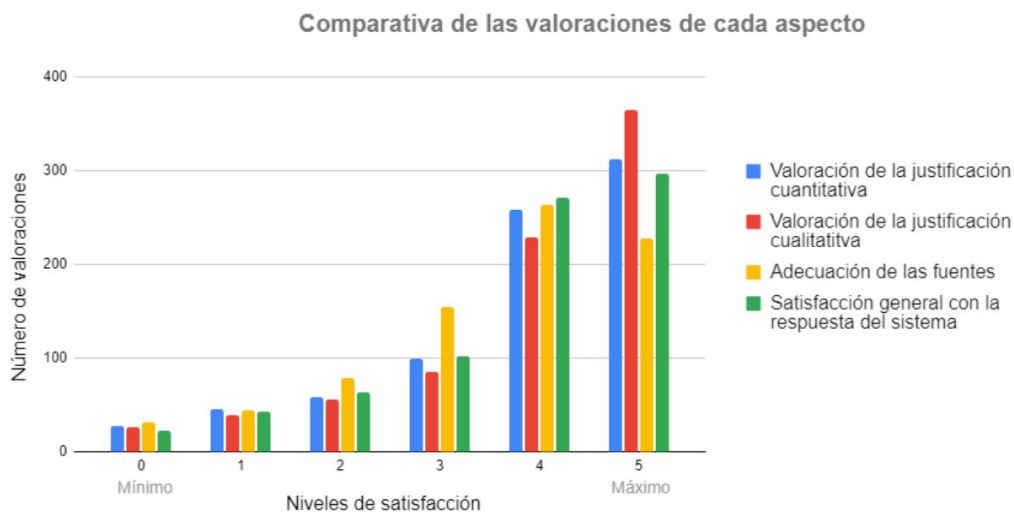


Figura 6.14: Comparativa de las valoraciones de cada aspecto del sistema

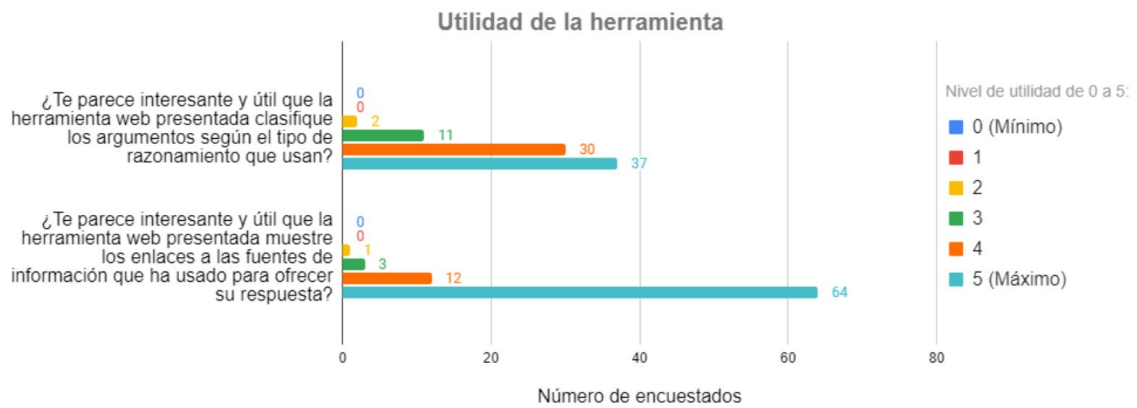
Los formularios también han incluido unas preguntas para evaluar el nivel de utilidad que los usuarios daban a dos de las características fundamentales y diferenciadoras del proceso de evaluación del sistema. Se ha preguntado:

- Si los encuestados consideraban interesante y útil que la herramienta web presentada clasifique los argumentos según el tipo de razonamiento que usan.
- Si los encuestados consideraban útil que la herramienta muestre los enlaces a las fuentes de información utilizadas para ofrecer su respuesta

Como se muestra en la Figura 6.15, la gran mayoría de los encuestados ha otorgado niveles satisfactorios de utilidad (3, 4 y 5) a ambas características, indicando que estas



funcionalidades proporcionan un alto valor añadido al sistema. En conclusión, esta respuesta positiva demuestra que los usuarios valoran tanto la clasificación de los argumentos por tipo de razonamiento como la transparencia al mostrar las fuentes de información, lo cual refuerza la relevancia y efectividad del sistema desarrollado.



**Figura 6.15:** Valoración de la utilidad del sistema



---

---

# CAPÍTULO 7

## Conclusiones

---

El presente trabajo ha supuesto el desarrollo exitoso de una herramienta diseñada para abordar el desafío crítico de detectar desinformación en argumentos expuestos por la sociedad, los cuales pueden manifestarse bajo diversas formas como falacias lógicas, manipulaciones retóricas o distorsiones de la verdad. Se han cumplido todos los objetivos establecidos al inicio del proyecto, presentados en la sección [1.2](#).

### 7.1 Cumplimiento de los objetivos

---

Se ha logrado desarrollar una herramienta web compuesta por dos módulos que combinan las técnicas de la argumentación computacional y el uso de grandes modelos de lenguaje. Por una parte, se ha implementado el módulo clasificador de esquemas argumentativos que logran clasificar de manera eficaz los argumentos en sus correspondientes esquemas argumentativos (OE2). Por otra parte, se ha desarrollado el módulo evaluador de la veracidad de los argumentos (OE3) utilizando un gran modelo de lenguaje con contextualización externa de diversas fuentes de información (OE4). En la evaluación final del sistema se presenta: una justificación cualitativa, una justificación cuantitativa y los enlaces a las fuentes utilizadas, aumentando la transparencia del sistema.

Además, se mejoró y optimizó el rendimiento de la herramienta final mediante la implementación de un sistema multitarea basado en programación concurrente (OE6), lo que contribuyó a una mayor eficiencia en la fase de recuperación de información de las distintas fuentes. También, se desarrolló una interfaz web (OE5) intuitiva para facilitar la interacción del usuario con la herramienta, permitiéndole introducir argumentos y recibir evaluaciones de manera clara y comprensible. Por último, la efectividad del sistema se evaluó a través del análisis de los resultados (OE7) obtenidos a partir de las evaluaciones cuantitativas y cualitativas llevadas a cabo, confirmando su capacidad para cumplir con su propósito de detectar desinformación de manera efectiva y promover el pensamiento crítico de los usuarios.

A lo largo del proyecto fue necesaria la realización de varios experimentos en cada módulo, con el objetivo de tomar las mejores decisiones informadas y determinar las estrategias correctas a seguir con el fin de conseguir implementar un sistema eficiente y eficaz. También fue determinante dedicar un largo periodo de tiempo a la lectura de numerosos artículos relacionados con el procesamiento del lenguaje natural, la argumentación computacional, la minería de argumentos, el uso y las limitaciones de los grandes modelos de lenguaje, y las técnicas de detección de desinformación más innovadoras. Esta exhaustiva revisión y análisis del estado del arte (OE1) permitió conocer los trabajos relacionados, así como familiarizarse con las tecnologías más usadas y las diferentes aproximaciones realizadas en otros proyectos, identificando sus ventajas e inconvenien-

tes. En definitiva, todo esto me permitió establecer una base sólida para el desarrollo y consecución de este proyecto.

## 7.2 Conocimientos adquiridos y dificultades encontradas

---

Gracias al desarrollo de este proyecto, he tenido la oportunidad de adquirir conocimientos significativos en varios aspectos. En primer lugar, exploré el campo de la argumentación computacional, área que desconocía totalmente y que he encontrado apasionante por su relevancia en el contexto actual debido a las limitaciones y desafíos que presentan los LLMs utilizados hoy en día. Por tanto, he podido profundizar en la teoría de la argumentación y en los diferentes patrones de razonamiento utilizados por los seres humanos al argumentar.

Además, aunque tenía cierta experiencia previa en tareas de procesamiento de lenguaje natural, nunca había abordado una tarea de esta magnitud. El desarrollo del sistema fue un desafío, ya que implicaba la integración de dos módulos distintos. El primero consistía en un clasificador de esquemas argumentativos a partir de texto natural, una tarea novedosa para mí, ya que estaba más familiarizada con clasificadores con otro tipo de entrada, como por ejemplo imágenes. Sin embargo, la aplicación y el aprendizaje del *framework* de Rasa fue fundamental para desarrollar este sistema clasificador de manera satisfactoria. Otro desafío importante fue diseñar un sistema clasificador compuesto por varios clasificadores combinados y organizados en dos capas.

Por otro lado, durante el desarrollo del segundo módulo del sistema, tuve la oportunidad de explorar y comparar diversos LLMs disponibles en la actualidad como LLAMA, Mistral y BERT, entre otros. Además, exploré diversas técnicas para potenciar un LLM como fue la experimentación con la librería LangChain. Aunque inicialmente esta opción parecía interesante, esta librería no se ajustaba a la necesidad de mantener una interacción adecuada entre el sistema final y el usuario, ya que su aplicación requería de un tiempo de ejecución elevado. Por tanto, opté por llevar a cabo una implementación desde cero en la que el LLM se potenciaba a través de la conexión directa a varias APIs de fuentes de información, logrando con esta opción una mejora significativa del temporal.

Finalmente, tuve que desarrollar una interfaz web e interconectar todos los módulos que utilizaban tecnologías completamente diferentes para crear el sistema final. Durante este proceso, descubrí la utilidad de la librería Streamlit para el desarrollo web, lo que me permitió conseguir implementar una interfaz de usuario adecuada para interactuar con el sistema.

En conclusión, el desarrollo de este trabajo ha supuesto una oportunidad para adquirir nuevos conocimientos y habilidades, así como para reflexionar sobre los desafíos y aprendizajes tanto profesionales como personales que conlleva la realización de un proyecto de esta envergadura.

## 7.3 Relación del trabajo desarrollado con los estudios cursados

---

Como estudiante de la rama de Computación en el Grado de Ingeniería Informática, para el desarrollo de este proyecto ha sido necesario aplicar muchos de los conocimientos aprendidos durante los estudios cursados. A continuación, se relaciona los conocimientos aplicados en el proyecto con diversas asignaturas cursadas:

- Conocimientos básicos de estructuras utilizadas en programación y lenguaje de programación Python: en las asignaturas de Algorítmica y Estructura de datos.

- Conocimientos sobre tareas de procesamiento de lenguaje natural: en Sistemas de almacenamiento y recuperación.
- Desarrollo de modelos clasificadores con su posterior evaluación siguiendo unas métricas determinadas: en Aprendizaje automático y Percepción.
- Conocimientos sobre la recuperación de información y su posterior procesamiento: en Sistemas de almacenamiento y recuperación y en Base de datos.
- Conocimiento sobre la paralelización de procesos y programación concurrente: en Computación paralela y Concurrencia y sistemas distribuidos.
- Desarrollo de interfaces amigables e intuitivas para mejorar la experiencia del usuario y facilitar la comunicación con el sistema: en Interfaces persona computador.
- Gestión general del proyecto siguiendo una planificación detallada: en Gestión de proyectos.

Por otra parte, se han puesto en prácticas las siguientes competencias transversales:

- **CT1 - Comprensión e integración:** fue fundamental comprender y sintetizar una amplia gama de conocimientos multidisciplinares, desde la teoría de la argumentación computacional hasta las técnicas avanzadas de procesamiento de lenguaje natural.
- **CT2 – Aplicación y pensamiento práctico:** el desarrollo de los módulos del sistema implicó aplicar conocimientos teóricos de manera práctica y experimental.
- **CT3 – Análisis y resolución de problemas:** durante el proyecto, se presentaron diversos desafíos técnicos y metodológicos. Fue necesario analizar estos problemas en profundidad y desarrollar soluciones innovadoras y efectivas.
- **CT4 - Creatividad, innovación y emprendimiento:** se fomentó la creatividad y la innovación, especialmente al combinar técnicas de argumentación computacional con procesamiento de lenguaje natural para abordar el problema de la desinformación.
- **CT5 – Diseño y proyecto:** el diseño del sistema implicó una planificación meticulosa y una estructuración detallada de cada módulo, desde la conceptualización inicial hasta la implementación final.
- **CT7 - Responsabilidad ética, medioambiental y profesional:** se consideró la responsabilidad ética al abordar la problemática de la desinformación, una cuestión de gran relevancia social.
- **CT9 - Pensamiento crítico:** el desarrollo de la herramienta exigió una constante evaluación crítica de las técnicas y metodologías empleadas, tomando decisiones informadas en cada momento.
- **CT10 - Problemas contemporáneos:** la desinformación es uno de los problemas más importantes en la sociedad actual y este proyecto lo aborda directamente.
- **CT11 – Aprendizaje permanente:** el proyecto implicó un proceso continuo de aprendizaje y adaptación a nuevas tecnologías y métodos. Aprender sobre la teoría de la argumentación, las técnicas avanzadas de procesamiento de lenguaje natural y los modelos de lenguaje grande fue esencial para el desarrollo del proyecto.

- **CT12 - Planificación y gestión del tiempo:** la complejidad del proyecto requirió una planificación cuidadosa y una gestión eficiente del tiempo a lo largo de todo su desarrollo.

## 7.4 Trabajos futuros

---

A continuación, se destacan las posibles mejoras que se podrían realizar en este proyecto de cara al futuro, así como aquellas líneas de trabajo que se considera mejor no seguir.

En primer lugar, se podría intentar ampliar las entradas del sistema a textos o noticias completas. Esto requeriría el desarrollo de un módulo para la detección de argumentos en un texto en lenguaje natural, una tarea completamente distinta y complementaria a la abordada en este proyecto.

Por otro lado, la contextualización externa realizada en este proyecto se ha basado en la técnica de *prompt engineering*. Se podría considerar otra técnica para mejorar el rendimiento del LLM, como el desarrollo de un sistema RAG, en el cual el modelo de lenguaje estaría conectado a una base de datos con información específica sobre la tarea. Sin embargo, esto podría implicar que el sistema desarrollado solo sea experto en un determinado dominio (el contenido de la base de datos) y no multidominio como el sistema actual. A pesar de esto, esta línea de investigación sigue siendo prometedora y se podría mantener abierta para futuras exploraciones.

Además, si se desea implementar una versión futura del sistema a gran escala, se podría considerar la opción de incorporar diferentes agentes inteligentes que permitirían una mayor flexibilidad y capacidad de gestión del sistema en un entorno más amplio y complejo. Por ejemplo, se podrían implementar agentes expertos en diferentes dominios o con acceso a diferentes fuentes de información y a través del uso de tecnologías de acuerdo concretar la respuesta final del sistema. De este modo y como se ha comentado anteriormente, se podría combinar el entrenamiento de varios agentes inteligentes con el desarrollo de sistemas RAG dirigidos a crear agentes expertos en dominios concretos que se comunicasen y llegasen a acuerdos sobre sus conclusiones.

No obstante, se resaltan dos líneas de trabajo que no se recomiendan seguir visto la experiencia de este proyecto. La primera es potenciar el LLM mediante *fine-tuning* con una base de datos específica, ya que esto requiere un gran coste computacional, como se comprobó en el experimento realizado con un LLM alojado localmente en un entorno virtual. Además, varios estudios [Gorur et al., 2024] demuestran que los resultados obtenidos en tareas de minería de argumentos mediante *fine-tuning* no tienden a superar los obtenidos con técnicas como *prompt engineering* o RAG.

La otra línea de trabajo que no se recomienda seguir es el uso de la librería LangChain en el sistema, ya que la abstracción de esta potente librería introduce un coste temporal demasiado alto que actualmente dificulta la adaptación a las necesidades del sistema.

# Bibliografía

---

- [Adoma et al., 2020] Adoma, A. F., Henry, N.-M., and Chen, W. (2020). Comparative analyses of bert, roberta, distilbert, and xlnet for text-based emotion recognition. In *2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pages 117–121. IEEE.
- [Astuti et al., 2021] Astuti, W., Putri, D. P. I., Wibawa, A. P., Salim, Y., Ghosh, A., et al. (2021). Predicting frequently asked questions (faqs) on the covid-19 chatbot using the diet classifier. In *2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT)*, pages 25–29. IEEE.
- [Ben-Kiki et al., 2009] Ben-Kiki, O., Evans, C., and Ingerson, B. (2009). Yaml ain't markup language (yaml<sup>TM</sup>) version 1.1. *Working Draft 2008*, 5(11).
- [Church, 2017] Church, K. W. (2017). Word2vec. *Natural Language Engineering*, 23(1):155–162.
- [de Sousa et al., 2024] de Sousa, L. H. H., Trajano, G., Morales, A. S., Sarkadi, S., and Panisson, A. R. (2024). Using chatbot technologies to support argumentation. In *16th International Conference on Agents and Artificial Intelligence (ICAART 2024)*. SciTePress.
- [El Baff et al., 2019] El Baff, R., Wachsmuth, H., Al Khatib, K., Stede, M., and Stein, B. (2019). Computational argumentation synthesis as a language modeling task. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 54–64.
- [Gillioz et al., 2020] Gillioz, A., Casas, J., Mugellini, E., and Abou Khaled, O. (2020). Overview of the transformer-based models for nlp tasks. In *2020 15th Conference on Computer Science and Information Systems (FedCSIS)*, pages 179–183. IEEE.
- [Goffredo et al., 2023] Goffredo, P., Espinoza, M., Villata, S., and Cabrio, E. (2023). Argument-based detection and classification of fallacies in political debates. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 11101–11112. Association for Computational Linguistics.
- [Gorur et al., 2024] Gorur, D., Rago, A., and Toni, F. (2024). Can large language models perform relation-based argument mining? *arXiv preprint arXiv:2402.11243*.
- [Guthrie et al., 2006] Guthrie, D., Allison, B., Liu, W., Guthrie, L., and Wilks, Y. (2006). A closer look at skip-gram modelling. In *LREC*, volume 6, pages 1222–1225.
- [Hardeniya et al., 2016] Hardeniya, N., Perkins, J., Chopra, D., Joshi, N., and Mathur, I. (2016). *Natural language processing: python and NLTK*. Packt Publishing Ltd.
- [Hwang et al., 2021] Hwang, M.-H., Shin, J., Seo, H., Im, J.-S., and Cho, H. (2021). Korasa: Pipeline optimization for open-source korean natural language understanding framework based on deep learning. *Mobile Information Systems*, 2021:1–9.

- [Iranzo-Sánchez and Ruiz-Dolz, 2019] Iranzo-Sánchez, J. and Ruiz-Dolz, R. (2019). Vrain at irosva 2019: Exploring classical and transfer learning approaches to short message irony detection. In *IberLEF@ SEPLN*, pages 322–328.
- [Jiang et al., 2023] Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al. (2023). Mistral 7b. *arXiv preprint arXiv:2310.06825*.
- [Khan et al., 2021] Khan, F. S., Al Mushabbir, M., Irbaz, M. S., and Al Nasim, M. A. (2021). End-to-end natural language understanding pipeline for bangla conversational agents. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 205–210. IEEE.
- [Kong et al., 2021] Kong, X., Wang, G., and Nichol, A. (2021). *Conversational AI with Rasa: Build, test, and deploy AI-powered, enterprise-grade virtual assistants and chatbots*. Packt Publishing Ltd.
- [Kotonya and Toni, 2019] Kotonya, N. and Toni, F. (2019). Gradual argumentation evaluation for stance aggregation in automated fake news detection. In *Proceedings of the 6th Workshop on Argument Mining*, pages 156–166.
- [Kudo and Richardson, 2018] Kudo, T. and Richardson, J. (2018). Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.
- [McBurney and Parsons, 2002] McBurney, P. and Parsons, S. (2002). Games that agents play: A formal framework for dialogues between autonomous agents. *Journal of logic, language and information*, 11:315–334.
- [Minaee et al., 2024] Minaee, S., Mikolov, T., Nikzad, N., Chenaghlu, M., Socher, R., Amatriain, X., and Gao, J. (2024). Large language models: A survey.
- [Mishra et al., 2022] Mishra, D. S., Agarwal, A., Swathi, B., and Akshay, K. C. (2022). Natural language query formalization to sparql for querying knowledge bases using rasa. *Progress in Artificial Intelligence*, 11(3):193–206.
- [Montoro-Montarroso et al., 2023] Montoro-Montarroso, A., Rosso, P., Panizo-Lledot, Á., Calvo-Figueras, B., Cantón-Correa, F. J., Chulvi, B., Huertas-Tato, J., Rementeria, M. J., and Gómez-Romero, J. (2023). Inteligencia artificial contra la desinformación: fundamentos, avances y retos.
- [Musi et al., 2023] Musi, E., Carmi, E., Reed, C., Yates, S., and O’Halloran, K. (2023). Developing misinformation immunity: How to reason-check fallacious news in a human-computer interaction environment. *Social Media+ Society*, 9(1):20563051221150407.
- [Nguyen et al., 2021] Nguyen, T. T., Le, A. D., Hoang, H. T., and Nguyen, T. (2021). Neu-chatbot: Chatbot for admission of national economics university. *Computers and Education: Artificial Intelligence*, 2:100036.
- [Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.
- [Qader et al., 2019] Qader, W. A., Ameen, M. M., and Ahmed, B. I. (2019). An overview of bag of words; importance, implementation, applications, and challenges. In *2019 international engineering conference (IEC)*, pages 200–204. IEEE.



- [Ruiz-Dolz et al., 2021] Ruiz-Dolz, R., Alemany, J., Barberá, S. M. H., and García-Fornes, A. (2021). Transformer-based models for automatic identification of argument relations: A cross-domain evaluation. *IEEE Intelligent Systems*, 36(6):62–70.
- [Ruiz-Dolz and Lawrence, 2023] Ruiz-Dolz, R. and Lawrence, J. (2023). Detecting argumentative fallacies in the wild: Problems and limitations of large language models. In *Proceedings of the 10th Workshop on Argument Mining*. Association for Computational Linguistics.
- [Ruiz-Dolz et al., 2024] Ruiz-Dolz, R., Taverner, J., Lawrence, J., and Reed, C. (2024). Nlas-multi: A multilingual corpus of automatically generated natural language argumentation schemes. *arXiv preprint arXiv:2402.14458*.
- [Shabbir et al., 2021] Shabbir, J., Arshad, M. U., and Shahzad, W. (2021). Nubot: Embedded knowledge graph with rasa framework for generating semantic intents responses in roman urdu. *arXiv preprint arXiv:2102.10410*.
- [Song et al., 2020] Song, X., Salcianu, A., Song, Y., Dopson, D., and Zhou, D. (2020). Fast wordpiece tokenization. *arXiv preprint arXiv:2012.15524*.
- [Stahl, 2006] Stahl, B. C. (2006). On the difference or equality of information, misinformation, and disinformation: A critical research perspective. *Informing Science*, 9:83.
- [Tay et al., 2022] Tay, L. Q., Hurlstone, M. J., Kurz, T., and Ecker, U. K. (2022). A comparison of prebunking and debunking interventions for implied versus explicit misinformation. *British Journal of Psychology*, 113(3):591–607.
- [Touvron et al., 2023] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. (2023). Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.
- [Vasiliev, 2020] Vasiliev, Y. (2020). *Natural language processing with Python and spaCy: A practical introduction*. No Starch Press.
- [Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.
- [Vijayarani et al., 2016] Vijayarani, S., Janani, R., et al. (2016). Text mining: open source tokenization tools-an analysis. *Advanced Computational Intelligence: An International Journal (ACII)*, 3(1):37–47.
- [Walton, 2005] Walton, D. (2005). *Fundamentals of Critical Argumentation*. Critical Reasoning and Argumentation. Cambridge University Press.
- [Walton et al., 2008] Walton, D., Reed, C., and Macagno, F. (2008). *Argumentation schemes*. Cambridge University Press.
- [Wang et al., 2017] Wang, Q., Xu, J., Chen, H., and He, B. (2017). Two improved continuous bag-of-word models. In *2017 international joint conference on neural networks (IJCNN)*, pages 2851–2856. IEEE.
- [Wolf et al., 2019] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. (2019). Huggingface’s transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

- [Yang et al., 2020] Yang, S., Yu, X., and Zhou, Y. (2020). Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example. In *2020 International workshop on electronic communication and artificial intelligence (IWECAI)*, pages 98–101. IEEE.
- [Yenduri et al., 2024] Yenduri, G., Ramalingam, M., Selvi, G. C., Supriya, Y., Srivastava, G., Maddikunta, P. K. R., Raj, G. D., Jhaveri, R. H., Prabadevi, B., Wang, W., et al. (2024). Gpt (generative pre-trained transformer)—a comprehensive review on enabling technologies, potential applications, emerging challenges, and future directions. *IEEE Access*.
- [Yin et al., 2017] Yin, W., Kann, K., Yu, M., and Schütze, H. (2017). Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*.

---

## APÉNDICE A

# Objetivos de Desarrollo Sostenible

---

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenible	Alto	Medio	Bajo	No procede
ODS 1. Fin de la pobreza.				X
ODS 2. Hambre cero.				X
ODS 3. Salud y bienestar.		X		
ODS 4. Educación de calidad.	X			
ODS 5. Igualdad de género.		X		
ODS 6. Agua limpia y saneamiento.				X
ODS 7. Energía asequible y no contaminante.				X
ODS 8. Trabajo decente y crecimiento económico.				X
ODS 9. Industria, innovación e infraestructuras.	X			
ODS 10. Reducción de las desigualdades.	X			
ODS 11. Ciudades y comunidades sostenibles.		X		
ODS 12. Producción y consumo responsables.				X
ODS 13. Acción por el clima.	X			
ODS 14. Vida submarina.			X	
ODS 15. Vida de ecosistemas terrestres.		X		
ODS 16. Paz, justicia e instituciones sólidas.	X			
ODS 17. Alianzas para lograr objetivos.			X	

Este trabajo se alinea con diversos Objetivos de Desarrollo Sostenible (ODS) de las Naciones Unidas. En particular, con los siguientes:

- ODS 4, sobre "*Educación de Calidad*", que tiene como objetivo garantizar una educación inclusiva y equitativa de calidad y promover oportunidades de aprendizaje durante toda la vida para todos. En este ámbito de actuación, la detección de desinformación promueve la alfabetización mediática, fomenta el pensamiento crítico y empodera los individuos para tomar decisiones informadas.
- ODS 9, sobre "*Industria, innovación e infraestructuras*", que tiene como objetivo promover la industrialización inclusiva y sostenible y fomentar la innovación tecnológica. Este proyecto se enmarca en el desarrollo de nuevas tecnologías y herramientas innovadoras para analizar y clasificar información de manera más eficiente y precisa.
- ODS 10, sobre "*Reducción de las desigualdades*", que busca reducir la desigualdad dentro de y entre los países. El trabajo realizado contribuye a este objetivo al aumentar y promover la accesibilidad a información veraz y de calidad, lo que puede ayudar a reducir la brecha de conocimiento entre grupos marginados o desfavorecidos.
- ODS 13, sobre "*Acción por el clima*", que tiene como objetivo tomar medidas para combatir el cambio climático y sus impactos. La desinformación puede afectar la percepción pública e información disponible sobre el cambio climático y obstaculizar la acción colectiva para abordar este problema global. En consecuencia, su detección contribuye a este ODS.
- ODS 16, sobre "*Paz, justicia e instituciones sólidas*", que busca promover sociedades pacíficas e inclusivas para el desarrollo sostenible, brindar acceso a la justicia para todos y construir instituciones eficaces, responsables e inclusivas a todos los niveles. Este trabajo contribuye a este objetivo al ayudar a identificar y contrarrestar la desinformación, que puede debilitar la confianza en las instituciones y socavar la estabilidad social y política.

Este trabajo aborda la relevancia de la educación de calidad y la alfabetización mediática en la sociedad contemporánea, lo que potencialmente capacita a individuos para tomar decisiones informadas y participar activamente en asuntos cívicos y democráticos. Además, al fomentar la detección y análisis de desinformación, contribuye a mejorar la transparencia y confianza en los medios de comunicación e instituciones públicas, lo que a su vez fortalece la democracia y el estado de derecho. En resumen, este trabajo final de grado tiene implicaciones significativas para varios Objetivos de Desarrollo Sostenible, como la promoción de la educación de calidad, la reducción de desigualdades, el fortalecimiento de instituciones, la acción por el clima y la promoción de la paz y la justicia, mostrando su contribución al logro de metas sostenibles a nivel global.

---

---

## APÉNDICE B

# GLOSARIO

---

- **Argumentación:** Proceso de presentar y estructurar razones o pruebas para apoyar o refutar una afirmación o posición.
- **Argumentación computacional:** Subcampo de la inteligencia artificial que estudia cómo las máquinas pueden procesar, generar y evaluar argumentos.
- **PLN (*Procesamiento de Lenguaje Natural*):** Área de la inteligencia artificial que se centra en la interacción entre las computadoras y los humanos a través del lenguaje natural. Incluye tareas como la comprensión, generación y traducción de texto.
- **LLM (*Large Language Model*):** Modelos de lenguaje de gran tamaño entrenados con grandes volúmenes de datos para comprender y generar texto. Algunos ejemplos son GPT, BERT, y LLAMA.
- **API (*Application Programming Interface*):** Conjunto de reglas y protocolos que permiten que diferentes aplicaciones se comuniquen entre sí.
- **Rasa:** Framework de código abierto utilizado para construir y personalizar chatbots y asistentes virtuales.
- **Streamlit:** Herramienta de código abierto utilizada para crear aplicaciones web interactivas y visualizaciones de datos de manera rápida y sencilla.
- **Contextualización externa:** Técnica que implica el uso de información adicional de diversas fuentes externas para enriquecer el análisis y mejorar la precisión de las evaluaciones realizadas por los modelos de lenguaje.
- **Prompt Engineering:** Proceso de diseñar y refinar las indicaciones o *prompts* utilizados para guiar el comportamiento de un modelo de lenguaje grande, optimizando así sus respuestas y rendimiento.
- **Sistema RAG (*Retrieval-Augmented Generation*):** Técnica que combina recuperación de información con generación de texto. El sistema recupera datos relevantes de una base de datos y los utiliza para mejorar la calidad y precisión de las respuestas generadas por el modelo de lenguaje.
- **Fine-tuning:** Proceso de ajustar un modelo de lenguaje previamente entrenado con datos adicionales específicos para mejorar su rendimiento en una tarea particular.