



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
(ICAI)

MÁSTER UNIVERSITARIO EN BIG DATA Y ANALÍTICA
AVANZADA

TRABAJO DE FIN DE MÁSTER

**Desarrollo de un sistema NLP: análisis temático,
resumen automático y RAG aplicado al corpus
web en enfermedades raras de newsRARE**

Autor:
Ignacio Hernández Bas

Director del proyecto:
Ana Laguna Pradas

10-04-2025

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Desarrollo de un sistema NLP: análisis temático, resumen automático y RAG aplicado al corpus web en enfermedades raras de newsRARE

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2025/2026 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos. El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Fdo.: Ignacio Hernández Bas Fecha: ...17./...06.../2025..



Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Ana Laguna Pradas Fecha: ...17./...06.../2025..





ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
(ICAI)

MÁSTER UNIVERSITARIO EN BIG DATA Y ANALÍTICA
AVANZADA

TRABAJO DE FIN DE MÁSTER

**Desarrollo de un sistema NLP: análisis temático,
resumen automático y RAG aplicado al corpus
web en enfermedades raras de newsRARE**

Autor:
Ignacio Hernández Bas

Director del proyecto:
Ana Laguna Pradas

10-04-2025

DESARROLLO DE UN SISTEMA NLP: ANALISIS TEMÁTICO, RESUMEN AUTOMÁTICO Y RAG APLICADO AL CORPUS WEB EN ENFERMEDADES RARAS DE NEWSRARE

Autor: Hernández Bas, Ignacio.

Director: Ana Laguna Pradas.

Entidad Colaboradora: newsRARE

RESUMEN DEL PROYECTO

Las ramas del NLP y la IA generativa avanzan con gran rapidez y, empleadas correctamente, aportan un gran valor y nuevas funcionalidades en los ámbitos sobre los que se aplican. En este proyecto se ha llevado a cabo el desarrollo de un sistema que facilite la comprensión y divulgación de contenidos especializados en enfermedades raras. A partir de la creación de un dataset propio, los resultados tras la evaluación son satisfactorios, confirmando la utilidad y el gran potencial que presentan las técnicas empleadas en la mejora del acceso al conocimiento sobre enfermedades raras, resultando en una configuración fácilmente extrapolable a otros campos de conocimiento especializados.

Palabras clave: NLP, Topic Modeling, transformers, embeddings, LLM, Chatbot, RAG,

1. Introducción

Actualmente, el desarrollo de técnicas de procesamiento del lenguaje natural (NLP) [1] ha experimentado un gran crecimiento, esto se ha producido en gran medida a la aparición de modelos basados en arquitecturas transformer [2]. Estos modelos han revolucionado la forma en la que las máquinas procesan, comprenden y generan lenguaje, habilitando la aparición de nuevas aplicaciones y funcionalidades. Entre las propuestas más al alza, se encuentra el uso de sistemas híbridos conocidos como RAG (Retrieval-Augmented-Generation) [3]. Estos sistemas combinan el poder generativo de modelos LLM con altas capacidades tales como GPT o BERT [4] con una base de datos vectorizada (embeddings) [5] propia la cual permite generar respuestas fundamentadas en el contexto deseado, mejorando la trazabilidad de las respuestas y reduciendo las alucinaciones generadas por el modelo.

Estos avances son especialmente relevantes en contextos donde la información que se dispone es muy extensa, dispersa o muy técnica. El ámbito de las enfermedades raras constituye uno de estos casos, donde se surge un desafío tanto para los profesionales sanitarios como para los divulgadores científicos. Esto se debe debido a la baja prevalencia de estas patologías y a la escasa dispersión del conocimiento disponible. La revista **newsRARE** trata de proponer una solución ante esta situación acercando estos conocimientos en España, aunque su volumen creciente plantea una serie de retos en cuanto a la organización y accesibilidad a esta información.

2. Definición del proyecto

Este proyecto define la creación de un sistema que permita acercar al usuario múltiples de las funcionalidades y herramientas relativas al procesamiento del lenguaje natural aplicadas al dominio de las enfermedades raras. Por lo tanto, se va a diseñar una aplicación que facilite la comprensión, organización y consulta de contenidos especializados mediante técnicas de extracción temática, resumen automático y generación aumentada por recuperación (RAG).

El objetivo es permitir que tanto profesionales como usuarios no especializados puedan acceder fácilmente a la información publicada en la revista newsRARE, mejorando así la

difusión del conocimiento en un ámbito poco accesible. Este sistema está enfocado principalmente en el tratamiento automático de textos, los cuales serán procesados para que puedan ser explorados temáticamente, resumidos y consultados a través de un asistente conversacional (chatbot) diseñado específicamente para este propósito.

3. Descripción del sistema

El sistema desarrollado se divide en tres bloques fundamentales:

- **Extracción de temas:** bloque encargado del análisis completo de todo el corpus documental de newsRARE para la extracción de las temáticas más representativas. Para ello se emplean técnicas Topic Modeling basadas en TF-IDF y NMF, junto a diferentes visualizaciones de los datos que permitirán la identificación de temas de forma intuitiva. El objetivo principal de este bloque se basa en ofrecer una organización estructurada del contenido de la web, permitiendo detectar patrones temáticos en el conjunto de publicaciones. Los temas extraídos se podrán añadir como etiquetas a cada artículo para que posteriormente el usuario pueda filtrar el contenido a través de estas.
- **Generación de resúmenes:** módulo enfocado en la aplicación de modelos preentrenados para el resumen automático de textos. Se han empleado modelos con diferentes enfoques: uno generalista y otro biomédico, lo cual permitirá comparar su rendimiento ante textos reales. El objetivo de módulo se centra en mejorar la difusión de contenidos mediante versiones sintetizadas de los artículos a través de las redes sociales
- **Chatbot RAG:** mediante la creación de una base de datos vectorial compuesta por los embeddings generados a partir del corpus documental, el sistema permite la combinación de diferentes modelos generativos con el contexto formado por los artículos de esta base de datos para poder responder las diferentes preguntas que planteen los usuarios. Esta herramienta se ha generado siguiendo la estructura RAG[5], la cual se muestra en la Figura 1.

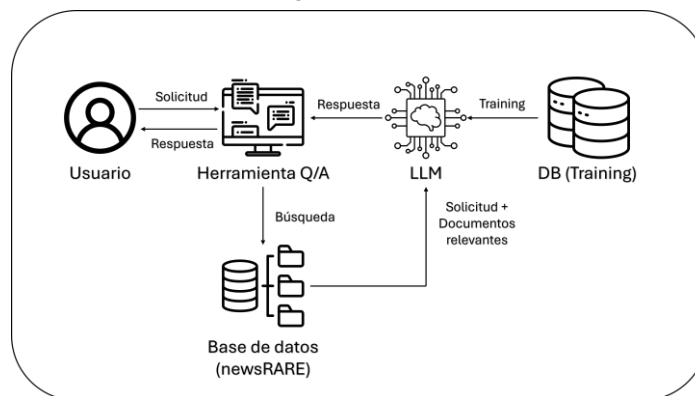


Figura 1 – Arquitectura del chatbot basada en el enfoque RAG

A partir de estos tres bloques se dispone de una solución que permite la transformación de un extenso corpus, formado por contenido desestructurado sobre una temática especializada, en un recurso más accesible y atractivo para el usuario. Además, el diseño modular permite la futura ampliación de las funcionalidades implementadas.

4. Resultados

Una vez implementado el sistema, es necesario evaluar el comportamiento de cada uno de los bloques principales:

- Extracción de temas:** se ha llevado a cabo una extracción de temáticas satisfactoria, permitiendo una obtención de tópicos bien diferenciada de los artículos (por ejemplo, temas como el cribado neonatal, la industria farmacéutica, evaluaciones tecnológicas) que forman parte del corpus newsRARE. Entre las diferentes combinaciones disponibles entre modelos, número de tópicos a extraer, y el tamaño de ellos n-gramas el modelo de extracción de temas con mejores resultados consiste un modelo NMF que emplea bigramas y extrae 15 tópicos, generando buenas métricas de coherencia (Figura 2) y generaron agrupaciones bien diferenciadas y espacialmente separadas, alineándose con los buenos resultados obtenidos en las métricas anteriores.



Figura 2 – Visualización de los 15 tópicos obtenidos empleando NMF y bigramas

- Generación de resúmenes:** se ha evaluado este módulo desde una perspectiva cuantitativa y cualitativa. El enfoque Medical (biomédico), que requiere traducción previa, obtiene mejores resultados en todas las métricas BERTScore, como por ejemplo la que se muestra en la Tabla 1. Además, en la evaluación humana, muestra generar resúmenes más completos, precisos y coherentes.

Método empleado	mean	std	min	max
General (mT5_multilingual_XLSum)	0.6943	0.0176	0.6611	0.7445
Medical (medical_summarization + traducción)	0.7431	0.0392	0.6437	0.8443

Tabla 1- Resultados evaluación (BERTScore-F1) de los resúmenes generados

- Chatbot RAG:** se ha evaluado este bloque atendiendo a los aspectos de veracidad, robustez, velocidad y escalabilidad. El modelo **BioGPT** junto a **bge-m3** ha sido el más preciso y coherente, superando en métricas a versiones más avanzadas. El sistema ha demostrado ser robusto frente a preguntas fuera de dominio y mantiene tiempos de respuesta adecuados. Además, la arquitectura basada en ChromaDB permite añadir nuevos documentos sin rehacer el sistema, asegurando su escalabilidad.

Pregunta	Modelo	ROUGE-L	BERTScore-F1	Cosine Similarity
Holoclar®	BioGPT	0.336	0.780	0.854
	BioGPT-Large	0.132	0.694	0.727
	Flan-T5-Large	0.134	0.697	0.266
	Flan-T5-XL	0.222	0.742	0.350
SpainRDR	BioGPT	0.243	0.678	0.448
	BioGPT-Large	0.115	0.644	0.531
	Flan-T5-Large	0.282	0.687	0.441
	Flan-T5-XL	0.167	0.660	0.481

Tabla 2- Métricas obtenidas por modelo para dos preguntas del dataset

5. Conclusiones

Gracias a la creación de este proyecto, se ha podido desarrollar un sistema completo que acerca al usuario información útil sobre un dominio específico como el de las enfermedades raras, permitiendo organizar, resumir y consultar contenido de forma automatizada. Esto abre una nueva vía para mejorar la accesibilidad y divulgación de conocimiento biomédico en español, así como en otros dominios especializados donde la aplicación de herramientas de procesamiento natural del lenguaje se consideraba especialmente compleja.

Los resultados obtenidos en cada uno de los bloques han sido satisfactorios, demostrando que el sistema funciona correctamente. Aun así, es importante señalar que el sistema podría seguir mejorando, combatiendo limitaciones tales como los recursos computacionales disponibles o la falta de un dataset de referencia evaluado por expertos en los artículos, lo cual permitiría obtener unas conclusiones más precisas respecto a la precisión y utilidad del sistema en escenarios reales.

6. Referencias

- [1] K. R. Chowdhary. “Natural Language Processing”. In: *Fundamentals of Artificial Intelligence*. Springer India, 2020, pp. 603–649. DOI: [10.1007/978-81-322-3972-7_19](https://doi.org/10.1007/978-81-322-3972-7_19).
- [2] Ashish Vaswani et al. “Attention is All You Need”. In: *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc., 2017.
- [3] Patrick Lewis et al. “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 9459–9474.
- [4] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).
- [5] Tomas Mikolov et al. “Efficient Estimation of Word Representations in Vector Space”. In: *arXiv preprint arXiv:1301.3781* (2013).
- [6] Manik Bhandari et al. “Re-evaluating Evaluation in Text Summarization”. In: *arXiv preprint arXiv:2010.07100* (2020).

DEVELOPMENT OF AN NLP SYSTEM: TOPIC MODELING, AUTOMATIC SUMMARIZATION, AND RAG APPLIED TO THE NEWSRARE WEB CORPUS ON RARE DISEASES

Author: Hernández Bas, Ignacio.

Director: Ana Laguna Pradas.

Collaborating Entity: newsRARE

ABSTRACT

The fields of NLP and generative AI are advancing rapidly and, when used correctly, they bring great value and new functionalities to the areas in which they are applied. This project has developed a system that facilitates the understanding and dissemination of specialized content on rare diseases. Based on the creation of a proprietary dataset, the results after evaluation are satisfactory, confirming the usefulness and great potential of the techniques used in improving access to knowledge about rare diseases, resulting in a configuration that can be easily extrapolated to other specialized fields of knowledge.

Keywords: NLP, Topic Modeling, transformers, embeddings, LLM, Chatbot, RAG,

1. Introduction

Currently, the development of natural language processing (NLP) techniques [1] has experienced significant growth, largely due to the emergence of models based on transformer architectures [2]. These models have revolutionized the way machines process, understand, and generate language, enabling the emergence of new applications and functionalities. Among the most promising proposals is the use of hybrid systems known as RAG (Retrieval-Augmented-Generation) [3]. These systems combine the generative power of LLM models with high capabilities such as GPT or BERT [4] with their own vectorized database (embeddings) [5], which allows them to generate responses based on the desired context, improving the traceability of responses and reducing the hallucinations generated by the model.

These advances are particularly relevant in contexts where the available information is extensive, scattered, or highly technical. The field of rare diseases is one such case, posing a challenge for both healthcare professionals and science communicators. This is due to the low prevalence of these diseases and the limited dissemination of available knowledge. The magazine newsRARE seeks to offer a solution to this situation by bringing this knowledge closer to Spain, although its growing volume poses several challenges in terms of the organization and accessibility of this information.

2. Project definition

This project defines the creation of a system that brings users multiple functionalities and tools related to natural language processing applied to the field of rare diseases. Therefore, an application will be designed to facilitate the understanding, organization, and consultation of specialized content through techniques such as topic extraction, automatic summarization, and retrieval-augmented generation (RAG).

The goal is to enable both professionals and non-specialized users to easily access the information published in the newsRARE magazine, thereby improving the dissemination of knowledge in a field that is not easily accessible. This system focuses primarily on the automatic processing of texts, which will be processed so that they can be explored thematically, summarized, and consulted through a conversational assistant (chatbot) designed specifically for this purpose.

3. System Description

The developed system is divided into three fundamental blocks:

- **Topic extraction:** block responsible for the complete analysis of the entire newsRARE document corpus to extract the most representative topics. To do this, Topic Modeling techniques based on TF-IDF and NMF are used, together with different data visualizations that will allow topics to be identified intuitively. The main objective of this block is to provide a structured organization of the website's content, allowing thematic patterns to be detected in the set of publications. The extracted topics can be added as tags to each article so that the user can later filter the content through them.
- **Summary generation:** module focused on the application of pre-trained models for automatic text summarization. Models with different approaches have been used: one generalist and one biomedical, which will allow their performance to be compared with real texts. The objective of the module is to improve the dissemination of content through synthesized versions of articles via social media.
- **RAG chatbot:** by creating a vector database composed of embeddings generated from the document corpus, the system allows the combination of different generative models with the context formed by the articles in this database in order to answer the different questions posed by users. This tool has been generated following the RAG[5] structure, which is shown in Figure 1.

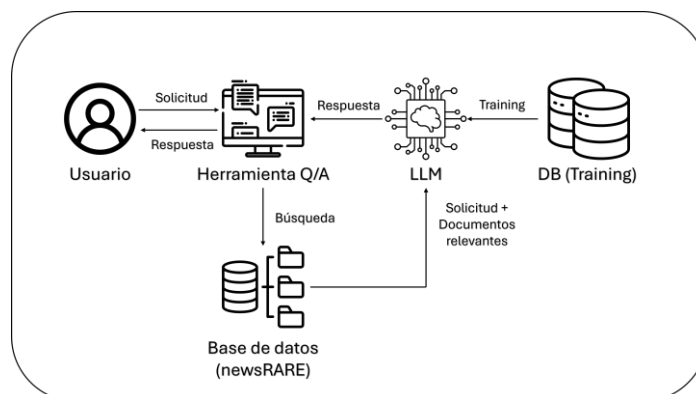


Figure 1 – Chatbot architecture based on the RAG approach

These three blocks provide a solution that allows the transformation of an extensive corpus, consisting of unstructured content on a specialized subject, into a more accessible and attractive resource for the user. In addition, the modular design allows for future expansion of the implemented functionalities.

4. Results

Once the system has been implemented, it is necessary to evaluate the behavior of each of the main blocks:

- **Topic extraction:** a satisfactory topic extraction has been carried out, allowing for the obtaining of well-differentiated topics from the articles (e.g., topics such as neonatal screening, the pharmaceutical industry, technology assessments) that form part of the newsRARE corpus. Among the different combinations available between models, number of topics to extract, and size of n-grams, the topic extraction model with the best results is an NMF model that uses bigrams and extracts 15 topics, generating good coherence metrics (Figure 2) and producing well-differentiated and spatially separated clusters, in line with the good results obtained in the previous metrics.



Figure 2 – Visualization of the 15 topics obtained using NMF and bigrams

- **Generación de resúmenes:** this module has been evaluated from a quantitative and qualitative perspective. The Medical (biomedical) approach, which requires prior translation, obtains better results in all BERTScore metrics, such as the one shown in Table 1. In addition, in human evaluation, it shows that it generates more complete, accurate, and coherent summaries..

Método empleado	mean	std	min	max
General (mT5_multilingual_XLSum)	0.6943	0.0176	0.6611	0.7445
Medical (medical_summarization + traducción)	0.7431	0.0392	0.6437	0.8443

Tabla 1- Evaluation results (BERTScore-F1) of the generated summaries

- **Chatbot RAG:** this block has been evaluated in terms of accuracy, robustness, speed, and scalability. The BioGPT model together with bge-m3 has been the most accurate and consistent, outperforming more advanced versions in terms of metrics. The system has proven to be robust against out-of-domain questions and maintains adequate response times. In addition, Chroma DB-based architecture allows new documents to be added without rebuilding the system, ensuring its scalability.

Pregunta	Modelo	ROUGE-L	BERTScore-F1	Cosine Similarity
Holoclar®	BioGPT	0.336	0.780	0.854
	BioGPT-Large	0.132	0.694	0.727
	Flan-T5-Large	0.134	0.697	0.266
	Flan-T5-XL	0.222	0.742	0.350
SpainRDR	BioGPT	0.243	0.678	0.448
	BioGPT-Large	0.115	0.644	0.531
	Flan-T5-Large	0.282	0.687	0.441
	Flan-T5-XL	0.167	0.660	0.481

Table 2- Metrics obtained by model for two questions from the dataset

5. Conclusiones

Thanks to the creation of this project, it has been possible to develop a comprehensive system that provides users with useful information on a specific domain such as rare diseases, allowing content to be organized, summarized, and consulted in an automated manner. This opens up a new avenue for improving the accessibility and dissemination of biomedical knowledge in Spanish, as well as in other specialized domains where the application of natural language processing tools was considered particularly complex.

The results obtained in each of the blocks have been satisfactory, demonstrating that the system works correctly. Even so, it is important to note that the system could still be improved by addressing limitations such as the available computational resources or the lack of a reference dataset evaluated by experts in the articles, which would allow for more accurate conclusions regarding the precision and usefulness of the system in real scenarios.

6. References

- [1] K. R. Chowdhary. “Natural Language Processing”. In: *Fundamentals of Artificial Intelligence*. Springer India, 2020, pp. 603–649. DOI: [10.1007/978-81-322-3972-7_19](https://doi.org/10.1007/978-81-322-3972-7_19).
- [2] Ashish Vaswani et al. “Attention is All You Need”. In: *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc., 2017.
- [3] Patrick Lewis et al. “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks”. In: *Advances in Neural Information Processing Systems 33 (2020)*, pp. 9459–9474.
- [4] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *arXiv preprint arXiv:1810.04805 (2018)*.
- [5] Tomas Mikolov et al. “Efficient Estimation of Word Representations in Vector Space”. In: *arXiv preprint arXiv:1301.3781 (2013)*.
- [6] Manik Bhandari et al. “Re-evaluating Evaluation in Text Summarization”. In: *arXiv preprint arXiv:2010.07100 (2020)*.

Índice

1. Introducción	13
2. Estado de la cuestión	14
2.1. Procesamiento del Lenguaje Natural (NLP), evolución e implementaciones	14
2.2. Funcionamiento de los modelos <i>NLP</i> y el empleo de técnicas RAG	15
2.3. Principales limitaciones en el ámbito del NLP	15
3. Definición del trabajo	18
3.1. Motivación	18
3.2. Objetivos del proyecto	19
3.3. Metodología de Trabajo y Planificación	20
4. Descripción de las tecnologías	23
4.1. Lenguaje de programación	23
4.2. Herramientas de <i>Machine, Deep learning</i> y <i>NLP</i>	23
4.3. Herramientas de visualización e interfaces	24
4.4. Otras herramientas complementarias	25
5. Sistema desarrollado	27
5.1. Análisis del sistema	27
5.2. Diseño e implementación del sistema	29
5.2.1. Scraping y procesamiento	29
5.2.1.1. Análisis del corpus	29
5.2.1.2. Diseño del scraper	31
5.2.1.3. Limpieza, estructuración y almacenamiento de los datos obtenidos	33
5.2.1.4. Consideraciones adicionales	34
5.2.2. Topic Modeling	34
5.2.2.1. Preprocesamiento del texto	34
5.2.2.2. Representación vectorial del texto	36
5.2.2.3. Modelado mediante NMF	37
5.2.2.4. Visualización y asignación de temas/tópicos	39
5.2.3. Automatización de resúmenes	39
5.2.3.1. Selección e implementación de modelos de resumen y traducción.	40
5.2.3.2. Generación de publicaciones	41
5.2.3.3. Diseño de la herramienta	41
5.2.4. Chatbot RAG (Retrieval Augmented Generation)	44
5.2.4.1. Creación de base de datos vectorial (embeddings)	45
5.2.4.2. Creación del dataset de preguntas y respuestas para la evaluación	46
5.2.4.3. Selección del modelo generativo	48
5.2.4.4. Implementación del RAG pipeline	48
6. Análisis de los resultados	51
6.1. Análisis de los temas extraídos	51
6.1.1. Evaluación cuantitativa	51

ÍNDICE

6.1.2. Evaluación cualitativa	55
6.2. Evaluación visual de los temas extraídos	56
6.3. Evaluación resúmenes generados	59
6.3.1. Evaluación cuantitativa	59
6.3.2. Evaluación cualitativa	61
6.4. Evaluación chatbot	64
6.4.1. Veracidad	64
6.4.1.1. Análisis cuantitativo	64
6.4.1.2. Análisis cualitativo	66
6.4.2. Robustez	70
6.4.3. Velocidad	72
6.4.4. Escalabilidad	74
6.4.5. Conclusión de la evaluación del sistema RAG	75
7. Conclusiones, limitaciones y trabajos futuros	76
7.1. Limitaciones halladas durante el proyecto	76
7.2. Líneas de trabajo futuro	77
ANEXO A: Detalles de implementación y código	80
A.1 Implementación Scraper	80
A.2 Implementación Topic Modeling	84
A.3 Implementación Resumen Automatizado	87
A.4 Implementación Chatbot RAG	89

Índice de figuras

3.1.	Diagrama de Gantt del proyecto	22
4.2.	Ejemplo de un gráfico interactivo creado con <i>Plotly</i>	25
4.3.	Ejemplo de un DataFrame creado en <i>Pandas</i>	26
5.4.	Visión conceptual del proyecto	29
5.5.	Página principal newsRARE	30
5.6.	Repositorio de artículos newsRARE	31
5.7.	Inspección de código fuente en navegador (Chrome)	32
5.8.	DataFrame obtenido tras el scraping web del repositorio de artículos	33
5.9.	WordCloud análisis exploratorio	35
5.10.	WordCloud tras eliminación de <i>stopwords</i>	35
5.11.	Secuencia lógica para la traducción/resumen de artículos	40
5.12.	Interfaz herramienta de resumen automatizado	42
5.13.	Ejemplo de resultados obtenidos mediante herramienta de resumen automatizado	43
5.14.	Arquitectura RAG implementada	44
5.15.	Interfaz del chatbot RAG (Recién iniciada la aplicación)	49
5.16.	Interfaz del chatbot RAG (Ejemplo de uso)	50
6.17.	Coherencia C_v en función del nº de topics y tamaño de n-grama	52
6.18.	Coherencia C_{umass} en función del nº de topics y tamaño de n-grama	53
6.19.	Asignación de tópicos en el corpus newsRARE	55
6.20.	Fragmento entrevista newsRARE	56
6.21.	t-SNE empleando unigramas y 15 topics	57
6.22.	t-SNE empleando bigramas y 15 topics	57
6.23.	t-SNE empleando trigramas y 15 topics	58
6.24.	Comparativa BERTScore obtenida entre modelos (General vs Médico)	60
6.25.	Noticia empleada para el análisis cualitativo de los resúmenes generados	62
6.26.	Métricas obtenidas evaluación por LLM y retriever	65
6.27.	Comparativa de robustez del chatbot RAG (% de preguntas correctamente detectadas)	71
6.28.	Tiempo medio (s) por modelo y retriever empleado	72
6.29.	Tiempo (s) por pregunta, modelo y retriever empleado	73

Índice de tablas

5.1. Comparación entre Stemming y Lematización	36
5.2. Ejemplo de extracción de n -gramas a partir de texto	37
5.3. Muestra de dataset de preguntas para los volúmenes publicados por newsRARE .	47
5.4. Comparativa de modelos generativos evaluados	48
6.5. Palabras clave de los tópicos 0, 5, 6 y 11 extraídos mediante NMF	55
6.6. Ejemplos representativos de temas extraídos por modelo de n -gramas	58
6.7. Métrica Bertscore Precision por modelo	60
6.8. Métrica Bertscore Recall por modelo	60
6.9. Métrica Bertscore F1 por modelo	60
6.10. Rúbrica para la evaluación cualitativa de resúmenes automáticos	61
6.11. Evaluación cualitativa de los modelos de resumen automático	63
6.12. Comparación de métricas de evaluación por modelo para las dos preguntas seleccionadas del conjunto de preguntas.	67
6.13. Comparación de respuestas generadas por los modelos para la pregunta sobre el medicamento Holoclar®	67
6.14. Comparación de respuestas generadas por los modelos para la pregunta sobre el proyecto SpainRDR	68

1. Introducción

En la actualidad, el ser humano dispone de una gran variedad de medios para acceder a la información que necesita. Con el auge de las tecnologías digitales, la sociedad contemporánea se encuentra inmersa en un entorno donde la accesibilidad a la información es más sencilla y rápida que nunca. Esta explosión de datos, generada por millones de dispositivos conectados, plantea nuevos desafíos para los métodos tradicionales de procesamiento de la información. Ante esta necesidad, emergen disciplinas como el *Big Data* y la *Inteligencia Artificial (IA)*, dentro de la cual se destaca especialmente el *Procesamiento del Lenguaje Natural (NLP)* y, más recientemente, la *IA generativa* como aquellos ámbitos en auge durante los últimos años.

El *NLP*[1] es la rama de la IA encargada de permitir que los ordenadores comprendan, interpreten y generen lenguaje humano. Gracias al uso de algoritmos avanzados y técnicas de aprendizaje automático, el NLP ha hecho posible tareas como la traducción automática, la clasificación de textos, la extracción de información y la generación automática de resúmenes, entre muchas otras.

Un punto de inflexión en la evolución del NLP y la IA generativa fue la aparición de la arquitectura *Transformer*[2], que reemplazó a los modelos secuenciales tradicionales como las redes neuronales recurrentes (RNN) y las LSTM. Esta nueva arquitectura, basada en mecanismos de atención, permitió un entrenamiento más eficiente y paralelo de los modelos de lenguaje, así como una mejor captura de dependencias a largo plazo en el texto. Gracias a esta innovación, fue posible desarrollar modelos de lenguaje cada vez más potentes y escalables, lo que marcó el inicio de la actual era de los *Large Language Models (LLMs)*. Estos modelos, entrenados con enormes volúmenes de texto, han dado paso a lo que hoy se conoce como *IA generativa*, una rama que permite a las máquinas generar contenido original, coherente y contextualizado en forma de texto, código, imágenes o incluso música mediante tecnologías como ChatGPT[3], Dall-e[4] y Suno AI[5], respectivamente.

La *IA generativa*, representada por modelos como GPT o BERT en el campo textual, ha revolucionado áreas tan diversas como la educación, la asistencia médica, el periodismo, la creatividad digital y la investigación científica. En particular, estos modelos han mostrado una notable capacidad para realizar tareas complejas como responder preguntas, redactar artículos, resumir documentos extensos, traducir entre múltiples idiomas o generar conversaciones fluidas en tiempo real. A medida que la potencia de cómputo continúa creciendo y los algoritmos se refinan, el acceso a estas tecnologías se ha democratizado, permitiendo su integración en aplicaciones de uso cotidiano. La conjunción del *NLP* y la *IA generativa* no solo redefine la interacción entre humanos y máquinas, sino que abre nuevas oportunidades para automatizar y mejorar procesos en prácticamente todos los sectores del conocimiento.

En este contexto, el presente proyecto tiene como objetivo desarrollar un chatbot especializado en *EERR*, capaz de proporcionar información clara, precisa y accesible a partir de un corpus documental concreto, en este caso, la revista newsRARE. Mediante el uso de técnicas **RAG** (Retrieval Augmented Generation), el sistema podrá recuperar información relevante y generar respuestas contextualizadas en lenguaje natural, facilitando el acceso a la información a usuarios no expertos.

2. Estado de la cuestión

A lo largo de este capítulo se va a exponer el estado del arte del *NLP* y la *IA Generativa* y sus múltiples aplicaciones en el procesamiento de textos y generación de respuestas. Para comenzar, se abordará el contexto en el que nos encontramos respecto a los usos e implementaciones de estas tecnologías. Seguidamente, se tratará acerca de los modelos encargados de llevar a cabo el procesamiento y su funcionamiento. Por último, se abordarán las limitaciones actuales de las implementaciones y se propondrá una nueva solución.

2.1. Procesamiento del Lenguaje Natural (NLP), evolución e implementaciones

En la sección anterior se introdujo el concepto de *NLP*, siendo la técnica computacional que permite a las máquinas comprender, interpretar y generar lenguaje humano. Los orígenes en este ámbito se remontan a la década de los 50, en el contexto de la Guerra Fría, donde se llevaron a cabo los primeros intentos en la traducción automática de documentos rusos al inglés. Dicho proyecto, implementado por la Universidad de Georgetown e IBM, marcó el comienzo de una disciplina la cual ha evolucionado exponencialmente en las siguientes décadas.

Durante las dos siguientes décadas se trató de enfocar el procesamiento del lenguaje de forma simbólica, estableciendo reglas gramaticales y lingüísticas que, en la práctica, no resultaron ser escalables con textos reales debido a la inherente complejidad del lenguaje humano. En los años 80 y 90, se dio el auge en el empleo de métodos estadísticos y probabilísticos de forma exitosa. Modelos tales como los modelos ocultos de Markov, para tareas de reconocimiento del habla y de etiquetado gramatical.

En 2013, surgió una gran revolución con la llegada de los *word embeddings*, como por ejemplo **Word2Vec**[6], los cuales permiten la representación de palabras como vectores en un espacio semántico, mejorando la capacidad de los modelos en la obtención de las relaciones sintácticas y semánticas de las palabras.

Más tarde aparecerían las redes neuronales recurrentes **RNNs** seguidas de las arquitecturas **Transformer**[2], las cuales provocarían la llegada de grandes modelos de lenguaje, como BERT[7] y GPT[3], permitiendo la compresión de textos y generación de texto coherente. Dichos modelos han sido entrenados con inmensos volúmenes de texto y han demostrado excelentes resultados en tareas de clasificación, resumen automático de textos y traducción, entre otras.

Actualmente, el *NLP* está integrado en múltiples aplicaciones cotidianas, desde los asistentes virtuales (como Siri o Alexa), a los sistemas de recomendación de contenidos, los motores de búsqueda y chatbots de asistencia. Por ejemplo, en el ámbito médico, se utiliza para extraer información clínica de historias médicas electrónicas, facilitar el diagnóstico o responder a consultas de pacien-

tes, lo que lo convierte en una herramienta esencial en la transformación digital del sector salud [8].

2.2. Funcionamiento de los modelos *NLP* y el empleo de técnicas RAG

Los modelos modernos de *NLP* se basan principalmente en arquitecturas de aprendizaje profundo, siendo la arquitectura Transformer la más influyente en los últimos años. Estos modelos están diseñados para capturar las relaciones semánticas y sintácticas en el texto gracias a mecanismos como la atención, que permite a cada palabra del texto tener en cuenta todas las demás en una secuencia. Además de que emplean procesamiento en paralelo de las palabras, aumentando su eficiencia computacional.

Sin embargo, los modelos generativos puros tienen una limitación crítica: su conocimiento está restringido al momento de su entrenamiento y no pueden acceder directamente a información externa actualizada o especializada (como artículos clínicos o bases de datos sobre enfermedades raras). Para superar esta limitación, se emplea una técnica llamada Retrieval-Augmented Generation (RAG) [9], la cual está basada en dos componentes fundamentales:

- **Retriever:** es un sistema que busca información relevante en una base de datos o corpus documental (por ejemplo, artículos médicos sobre enfermedades raras). Utiliza representaciones vectoriales (embeddings) y motores de búsqueda semántica (como ChromaDB) para recuperar los documentos más relacionados con una consulta del usuario.
- **Generator:** toma la pregunta del usuario junto con los documentos recuperados y genera una respuesta contextualizada utilizando un modelo generativo. Este componente se encarga de redactar una respuesta coherente, precisa y adaptada al contexto establecido por el retriever.

La combinación de estos elementos permite que el sistema sea más fiable, más escalable y especializado en los dominios deseados, lo cual es fundamental en el entorno médico, donde la precisión y la trazabilidad de las respuestas son clave, ya que se quiere evitar a toda costa alucinaciones por parte del modelo e información errónea o mal fundada.

2.3. Principales limitaciones en el ámbito del *NLP*

EL área del procesamiento natural del lenguaje o *NLP*, se encuentra en pleno auge, pero este ámbito no carece de limitaciones tanto técnicas como éticas, las cuales han de ser consideradas a la hora de llevar a cabo la implementación de estas.

Las principales **limitaciones técnicas** más destacadas son las siguientes:

- A nivel de los **datos**: existe una fuerte dependencia del corpus documental sobre el que se aplican las técnicas. La calidad, actualidad y cobertura temática del conjunto de datos influye directamente en la eficacia del sistema, especialmente en arquitecturas basadas en recuperación de información como *RAG*.
- A nivel de **modelo**: los modelos de lenguaje a gran escala *LLMs* actúan como cajas negras. A pesar de que son muy potentes, su interpretabilidad es muy baja, por lo que adaptarlos a aplicaciones especializadas supone una ardua tarea debido a la elevada complejidad y cantidad de parámetros de estos modelos. A su vez, el empleo de *LLMs* supone un alto coste computacional, lo que limita su accesibilidad a usuarios con recursos limitados, ya que su uso requiere hardware especializado como GPUs habilitadas o mediante el uso de entornos en la nube (por ejemplo, Google Colab) que ofrecen estos servicios de forma limitada.
- A nivel de la **recuperación** de documentos (**RAG**): los mecanismos de recuperación empleados en sistemas *Retrieval Augmented Generation* pueden ser imprecisos o introducir ruido. Es más, los modelos generativos pueden no ser capaces de utilizar adecuadamente la información recuperada, lo que afecta a la calidad de la respuesta final.
- A nivel de **evaluación**: la evaluación de tareas de generación abiertas, tales como la respuesta de preguntas o la generación de texto, supone una dificultad añadida, ya que las métricas de evaluación (BLEU, ROUGE o incluso BERTScore) no siempre reflejan de forma clara la calidad semántica, coherencia o factualidad de las respuestas generadas.

Por otro lado, se identifican las principales **limitaciones éticas** de los modelos empleados en el *NLP*, las cuales no pueden ser ignoradas a la hora de implementar soluciones con dichos modelos:

- **Alucinaciones**: Los modelos *NLP* son capaces de generar respuestas semánticamente correctas pero falsas en lo que a los hechos se refiere. Por lo que es importante que los modelos empleados deben de estar correctamente delimitados para responder factualmente con información contrastada para no caer en esta problemática, especialmente en contextos sensibles como la medicina, el derecho o la investigación científica.
- **Privacidad**: dado que los modelos pueden aprender patrones a partir de datos personales incluidos en los conjuntos de entrenamiento, es esencial garantizar que la privacidad de los individuos no se vea perjudicada. Esto implica cumplir con las normativas de protección de datos (GDPR[10]) y evitar que usuarios no autorizados dispongan de acceso a información confidencial por error.
- **Responsabilidad y trazabilidad**: la baja interpretabilidad de los modelos (caja negra) y la falta de mecanismos claros de trazabilidad dificultan el análisis de errores y la asignación de responsabilidades cuando se generan respuestas inadecuadas o dañinas.
- **Desigualdad en el acceso**: la elevada demanda computacional de los modelos de gran escala y la necesidad de grandes volúmenes de datos de calidad favorecen a las grandes corporaciones tecnológicas, generando una brecha entre estas y otras instituciones o grupos de

investigación con menos recursos, lo que limita la equidad en el desarrollo y acceso a estas tecnologías.

Aunque el avance en el campo del procesamiento del lenguaje natural y los modelos generativos ha supuesto un gran avance en numerosas aplicaciones, su implementación y uso conllevan importantes desafíos técnicos y éticos. Dificultades como el alto coste computacional, la obtención de respuestas factualmente verdaderas o la baja interpretabilidad de los modelos evidencian la necesidad de llevar a cabo aplicaciones y soluciones enfocadas en el tratamiento responsable de los datos y la adaptación correcta de dichas soluciones a cada contexto. Ante esto, este proyecto plantea una propuesta que busca implementar soluciones *NLP* y *RAG*, que prioricen tanto la fiabilidad de la información como la accesibilidad y transparencia del sistema en torno a un dominio especializado, como es el caso de las enfermedades raras.

3. Definición del trabajo

3.1. Motivación

El diagnóstico y tratamiento de enfermedades raras continúa siendo un desafío tanto para los profesionales sanitarios como para los propios pacientes, debido a la escasa información disponible, la dispersión del conocimiento y la baja prevalencia de estas patologías. En este contexto, surge la necesidad de emplear herramientas tecnológicas que acerquen el conocimiento médico especializado al público general, permitiendo una mayor comprensión y concienciación sobre estas enfermedades tan poco comunes.

A pesar de que este proyecto trabaje específicamente en el dominio clínico de las enfermedades raras, la metodología propuesta resulta igualmente aplicable a otros dominios, como el jurídico o el asegurador, donde la información que se dispone, al ser extensa, técnica y fragmentada, supone una barrera para el acceso al conocimiento por parte de usuarios no expertos.

El desarrollo de un chatbot especializado representa una solución innovadora y accesible para mitigar esta brecha de información. Gracias a la integración con una fuente especializada y verificada como la revista newsRARE, el chatbot no solo buscará ofrecer respuestas relevantes y actualizadas, sino que también lo hará en un lenguaje comprensible, facilitando así la divulgación científica y el empoderamiento de los pacientes y sus familias.

Además, el diseño del sistema tiene en cuenta las limitaciones computacionales que pueden suponer los modelos generativos de gran tamaño. Por ello, se apuesta por un enfoque basado en la técnica *RAG*, que permite utilizar modelos más ligeros sin sacrificar la coherencia y calidad de las respuestas. Esta estrategia busca alcanzar un equilibrio entre precisión, coste computacional y facilidad de implementación, lo cual resulta especialmente relevante para garantizar el acceso abierto y gratuito de la herramienta, favoreciendo así su despliegue en contextos educativos, sanitarios y sociales.

De forma complementaria, es de gran importancia para una página web que presenta una gran cantidad de datos, como es newsRARE, que se lleve a cabo la creación de un bloque de *Topic Modeling* que permita ofrecer un conocimiento más organizado, accesible y de utilidad. Este bloque permitirá identificar, agrupar y estructurar las temáticas más representativas halladas en los textos, convirtiéndose en una herramienta de gran valor para la propia revista, facilitando la organización de su propio contenido.

A su vez, con el fin de mejorar la accesibilidad del contenido hacia el público general, se va a llevar a cabo el diseño e implementación de un último bloque enfocado en la generación automática de resúmenes. Este módulo permitirá condensar la información de los artículos con el objetivo de que estos puedan ser fácilmente publicados en diferentes redes sociales. De esta forma, no solo se captará la atención de nuevos lectores hacia la plataforma digital, sino que se estará favoreciendo la divulgación de contenidos basados en las enfermedades raras.

3.2. Objetivos del proyecto

El *objetivo principal* de este proyecto es acercar a los usuarios los conocimientos relativos a enfermedades raras a partir de un **chatbot accesible** que permita al usuario interactuar de forma sencilla y rápida, otorgando respuestas relevantes, coherentes y reales, basándose en los conocimientos otorgados por la revista newsRARE.

Este proyecto plantea una serie de *retos*, por lo que es necesario establecer unos **objetivos** que sirvan como guía para comprobar que el desarrollo del mismo se lleva a cabo de forma satisfactoria. Se fijan los siguientes:

- **Veracidad:** el sistema debe generar respuestas basadas en información verificada y contenida en el corpus documental proporcionado. Para ello, se integrará un mecanismo de trazabilidad que permita mostrar los fragmentos del texto en los que se apoya la respuesta generada.
- **Robustez:** el chatbot debe basarse en la información documental que posee para evitar devolver al usuario información falsa, mal referenciada o causa de alucinación por parte del modelo. Además, deberá de saber cuando una pregunta realizada se encuentra fuera del caso de uso establecido, en este caso, generar respuestas en lo relativo al mundo de las enfermedades raras.
- **Velocidad:** el sistema debe ser lo suficientemente rápido para que la experiencia del usuario no se vea afectada. Por lo que tanto la carga de recursos como las respuestas por parte del modelo no deberán de superar los 2 minutos.
- **Escalabilidad:** la arquitectura del sistema debe permitir la integración de nuevos documentos o números adicionales de la revista *newsRARE* sin necesidad de reconstruir el sistema desde cero. Se comprobará que es posible indexar nueva información y regenerar los embeddings almacenados de forma incremental, manteniendo el rendimiento y sin afectar la precisión de las respuestas anteriores.

Además, se plantean unos *objetivos secundarios*, los cuales permitirán explorar y profundizar en el conocimiento contenido en los documentos que conforman el corpus de este proyecto, al tiempo que se aporta estructura y organización a la web:

- **Clasificación temática de los artículos:** identificar la temática principal de los artículos mediante la aplicación de técnicas de clasificación y *topic modeling*, facilitando así su organización y posterior análisis.
- **Resumen automático de contenidos:** automatizar el proceso de generación de resúmenes con el objetivo de fomentar la divulgación de los artículos y acercar información especializada al público general, especialmente a través de canales como redes sociales.

3.3. Metodología de Trabajo y Planificación

La metodología por la que se ha optado para el desarrollo de este proyecto recibe el nombre de metodología Agile, la cual se basa en un planteamiento iterativo, donde los resultados que se obtengan generarán cambios en el diseño o configuración de las soluciones planteadas. El proyecto se estructura en cuatro bloques principales, los cuales están organizados en subtareas.

En la figura anterior, se muestra un diagrama de Gantt en el cual se estima la duración de las tareas principales del proyecto. Cabe destacar que el desarrollo del proyecto va a seguir una metodología *Agile*, lo que implica que, a lo largo de la implementación de las tareas y en función de los resultados obtenidos, se llevarán a cabo cambios en el planteamiento del proyecto. El proyecto en sí mismo se divide en cuatro fases principales, las cuales están formadas por las siguientes subtareas:

1. Scraping y Análisis Temático del Corpus

1.1 Recopilación y limpieza del corpus

1.2 Análisis de tópicos con NMF

1.3 Visualización y evaluación de temas

2. Resumen Automático y Generación de Contenido

2.1 Implementación de modelos de resumen

2.2 Generación de dataset de resumen por artículo

2.3 Evaluación de calidad de los resúmenes

2.4 Generación automática de posts para redes sociales

3. Desarrollo del Chatbot RAG

3.1 Indexación de documentos y embeddings

3.2 Creación del dataset de preguntas y respuestas

3.3 Selección y pruebas de modelo generativo

3.4 Implementación del RAG pipeline

3.5 Evaluación automática de respuestas generadas e implementación de mejoras

4. Validación, Documentación y Entrega

4.1 Validación cualitativa de outputs

4.2 Revisión final y entrega

En la siguiente página se muestra la planificación de las tareas documentadas para el proyecto. El diagrama de **Gantt** empleado supone un enfoque de desarrollo muy tradicional (modelo *waterfall*), donde las tareas empleadas se realizaban de manera progresiva y, en cuanto estas finalicen, no se ve contemplada su modificación. Por ello, es importante recalcar que este diagrama es una representación organizativa de las tareas que se realizan en el proyecto, pero siguiendo la metodología Agile, muchas tareas han influido en otras previamente completadas, ajustándose de forma dinámica a los aprendizajes y resultados obtenidos en cada fase del proceso. Cabe destacar que la tarea de documentación del proyecto es continua y transcurre a lo largo de todas las fases del proyecto.

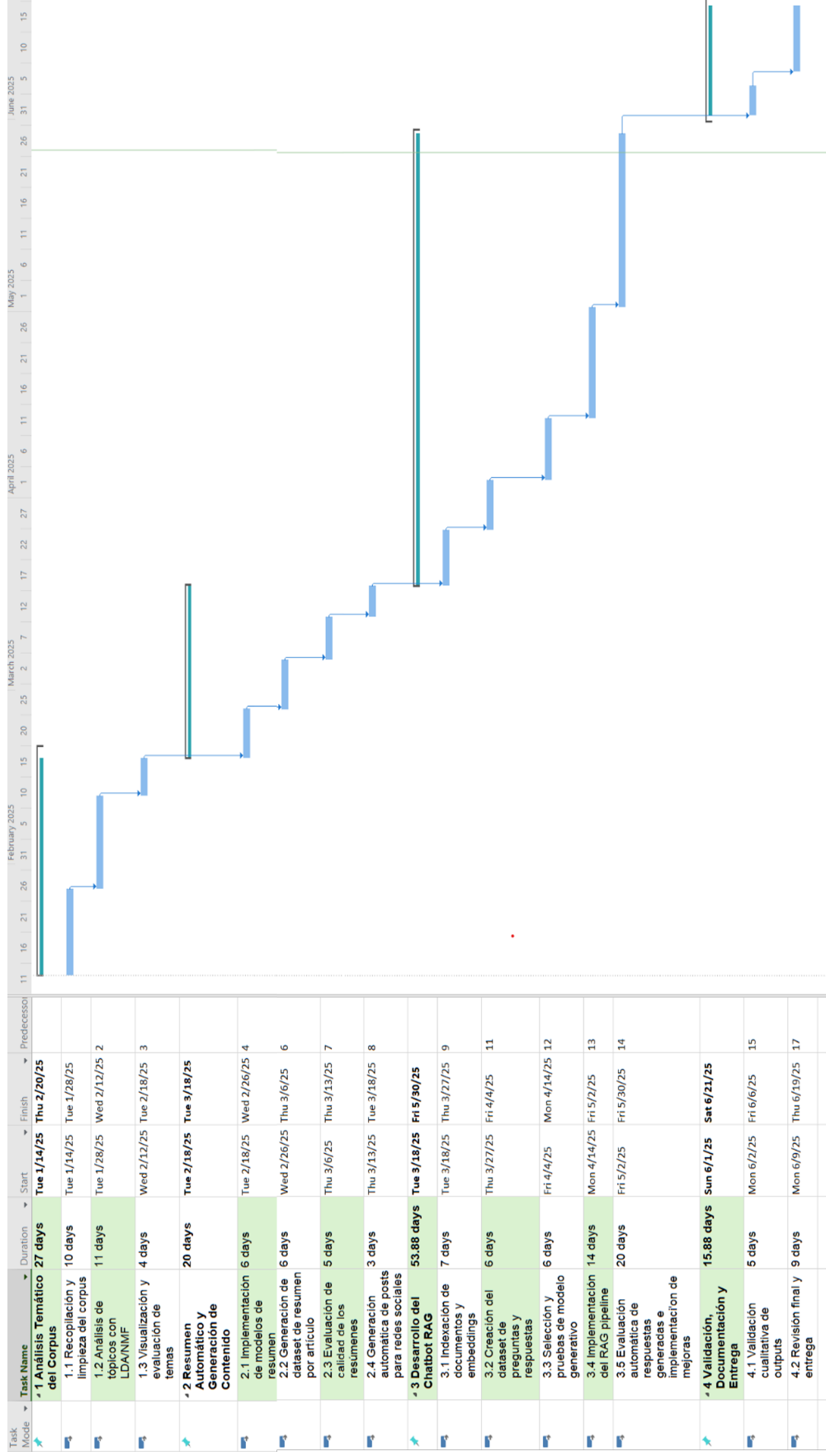


Figura 3.1: Diagrama de Gantt del proyecto

4. Descripción de las tecnologías

En este capítulo se lleva a cabo la descripción de las tecnologías utilizadas durante el desarrollo de este proyecto. A partir de la selección de dichas tecnologías, se define tanto el planteamiento como la implementación del trabajo. Este proyecto gira en torno al análisis de textos médicos mediante técnicas de *NLP*, la generación de contenido orientado a la difusión de información especializada a través de redes sociales y la creación de una aplicación interactiva basada en un *chatbot*. Por lo tanto, se han seleccionado aquellas tecnologías que permiten abordar eficazmente estas tareas, las cuales se detallan en la sección 3.2.

4.1. Lenguaje de programación

El lenguaje de programación que se utilice en el proyecto es fundamental, ya que todas las demás herramientas dependerán de la compatibilidad que estas muestren con dicho lenguaje. Además, el lenguaje seleccionado debe concordar de la mejor manera con la solución que se desea plantear.

Por estos motivos, el lenguaje seleccionado es *Python*, esto se debe a que es el lenguaje más popular en cuanto a análisis, procesamiento de datos y aplicación de técnicas relativas al *Machine y Deep Learning*. *Python* es un lenguaje que permite una fácil integración con librerías como *Gradio* o *Streamlit*, las cuales permiten la creación de aplicaciones sencillas sin la necesidad de uso de lenguajes de programación web como *Java*, *JavaScript*, *HTML* o *CSS*. Por estas capacidades, se ha optado por *Python* y no otros lenguajes más centrados en el análisis puro como *R*.

La capacidad de que todas las funcionalidades del proyecto funcionen a partir de un único lenguaje de programación permite una gran cohesión entre los diferentes elementos, favoreciendo la eficiencia durante el desarrollo del proyecto.

4.2. Herramientas de *Machine, Deep learning* y *NLP*

Una vez elegido el lenguaje de programación, se han seleccionado las librerías/herramientas que permiten el análisis de los textos mediante el uso de técnicas *NLP* avanzadas, tales como el resumen automático de textos o la implementación de una infraestructura *RAG*, posibilitando tanto la comprensión semántica de textos médicos como la generación de respuestas adaptadas al contexto ante preguntas del usuario.

- **Transformers (Hugging Face)**[11]: Librería base que proporciona acceso a modelos de lenguaje preentrenados como *BioMedLM* y *Meta-Llama-3-8B-Instruct*, fundamentales para tareas de generación y comprensión de lenguaje natural.
- **LangChain**[12]: Framework que permite integrar modelos LLM con fuentes externas como

bases de datos vectoriales, gestionando flujos conversacionales y facilitando la aplicación de arquitecturas RAG.

- **ChromaDB**[13]: Base de datos vectorial empleada para el almacenamiento y búsqueda semántica de fragmentos relevantes del corpus documental. Cada fragmento ha sido transformado en un vector utilizando modelos de *Sentence Transformers*.
- **SentenceTransformers**[14]: Librería para la generación de *embeddings* semánticos de texto, que permiten representar conceptos médicos en forma numérica, optimizando su recuperación por similitud.
- **Scikit-learn**[15]: Empleada para la implementación de técnicas de *topic modeling* como LDA o NMF, con el fin de identificar temas predominantes en los artículos y facilitar su organización.

4.3. Herramientas de visualización e interfaces

Es necesario disponer de herramientas que permitan la visualización de datos para observar el comportamiento de las diferentes funcionalidades del sistema.

- **Matplotlib**[16] y **Plotly**[17]: Utilizadas para la representación gráfica de métricas, análisis exploratorio del corpus, distribución de temas y evolución del rendimiento del sistema.

Matplotlib permite un gran grado de personalización, lo cual resulta muy funcional pero a su vez puede resultar compleja para los casos en los que se busque un método de representación más sencillo de implementar.

Por otro lado, *Plotly* está orientado a la generación de gráficos interactivos, los cuales pueden ser modificados por los usuarios en tiempo real, resultando en una opción muy atractiva para exponer la información.



Figura 4.2: Ejemplo de un gráfico interactivo creado con *Plotly*

- **Gradio**[18]: Framework que permite el desarrollo rápido de interfaces web ligeras en Python. Ha sido empleado para implementar una interfaz interactiva del chatbot, facilitando su prueba por parte de usuarios finales sin conocimientos técnicos. Además, puede ser alojada e implementada en repositorios propios de HuggingFace conocidos como *HuggingFace Spaces*.

4.4. Otras herramientas complementarias

Además de las tecnologías directamente relacionadas con el procesamiento del lenguaje natural y la visualización, se han empleado herramientas auxiliares que han facilitado tareas fundamentales en fases previas del desarrollo, como la recolección automatizada de los artículos, el procesamiento y limpieza de los datos obtenidos y el desarrollo de cálculos numéricos para la obtención de métricas.

- **Selenium**[19]: Librería orientada a la automatización de navegadores web, ampliamente utilizada en tareas de scraping y testing automatizado. En este proyecto, se ha empleado para la extracción automática de artículos desde el portal web de la revista *newsRARE*, simulando la navegación de un usuario y permitiendo la descarga estructurada del contenido relevante.
- **NumPy**[20]: Utilizada para realizar operaciones numéricas de bajo nivel, especialmente durante la etapa de evaluación y análisis de las métricas semánticas obtenidas de los textos generados. Esta biblioteca destaca por su alta eficiencia computacional, ya que está implementada en C, un lenguaje de bajo nivel que permite una gestión manual de la memoria,

reduciendo significativamente los tiempos de procesamiento y ejecución. Resulta especialmente útil para realizar operaciones vectorizadas y trabajar con matrices de múltiples dimensiones.

- **Pandas**[21]: Herramienta fundamental para la manipulación y transformación de datos en formato tabular. Extiende las funcionalidades de *NumPy* mediante estructuras avanzadas como los *DataFrames*, facilitando la carga, limpieza, organización y análisis de datos. Ha sido empleada en diversas etapas del proyecto, incluyendo el tratamiento del corpus, la creación de datasets de evaluación y la estructuración de los resultados obtenidos.

	source_json_file	title	content
0	noticias_articulos.json	HUMANIZACIÓN Y ATENCIÓN CENTRADA EN LA PERSONA...	CELIA GARCÍA MENÉNDEZDirectora General de Huma...
1	noticias_articulos.json	EU HTA Joint Clinical Assessment: are patients...	EU HTA JOINT CLINICAL ASSESSMENT. ARE PATIENTS...
2	noticias_articulos.json	OPTIMIZACIÓN DE LA PARTICIPACIÓN DE LOS PACIEN...	Zozaya, N., Fernández, I., Saldaña, R. Optimiz...
3	noticias_articulos.json	Carlos Dévora, Néboa Zozaya, Patricia Rizzi – ...	NUEVOS MARCOS EN EVALUACIÓN DE TECNOLOGÍAS SAN...
4	noticias_articulos.json	José Martínez Olmos – Patient mobility and hea...	Perna R, Cruz-Martínez G, Moreno Fuentes FJ. M...

Figura 4.3: Ejemplo de un DataFrame creado en *Pandas*

5. Sistema desarrollado

En esta sección se encuentra la descripción del sistema desarrollado. La estructura y arquitectura del sistema será analizada. Seguidamente, se describirá en mayor detalle el diseño de cada uno de los bloques principales que componen el proyecto para finalmente explicar el proceso de implementación de este.

5.1. Análisis del sistema

La solución planteada desde la visión de más alto nivel, tal como muestra la imagen 5.4, se puede ver como el sistema se encuentra segmentado en cuatro bloques fundamentales:

- **Scraping y Procesamiento:** punto de partida del proyecto, ya que a partir de él se lleva a cabo la obtención de toda la información que requieren los bloques posteriores. En este bloque se lleva a cabo el scraping web de la revista newsRARE, la limpieza y el correcto almacenamiento de cada uno de los artículos que presenta esta revista. Para ello es fundamental hacer un estudio riguroso del código fuente de la web origen para poder establecer sistemas automatizados que permitan la obtención de la información deseada de la forma más eficiente posible. Este bloque es fundamental para cumplir todos los objetivos planteados en el proyecto, el flujo de tareas necesarias para su correcta implementación es el siguiente:
 1. Análisis del corpus
 2. Diseño del scraper
 3. Limpieza, estructuración y almacenamiento de los datos obtenidos
- **Topic Modeling:** bloque analítico cuyo objetivo principal está centrado en la obtención de temáticas comunes a los artículos recopilados mediante técnicas de *NLP*. Este análisis no solo permite ofrecer diferentes posibilidades de catalogación del corpus obtenido, sino que permitirá además aportar nuevas visiones sobre el propio contenido disponible, como cuáles son las temáticas más utilizadas o de qué temas se están tratando menos, facilitando posteriores tareas de clasificación y generación de nuevos contenidos para la revista. Este bloque representa el primer **objetivo secundario** que se va a abordar en el proyecto y a continuación, se muestra el *pipeline* de tareas asignado:
 1. Preprocesamiento del texto
 2. Representación vectorial del texto

3. Modelado mediante NMF
 4. Visualización y asignación de temas/tópicos
- **Resumen RRSS:** apartado centrado en el resumen de los artículos de la página, permitiendo la generación automática de textos para ser publicados en redes sociales con el objetivo de acercar los conocimientos en enfermedades raras a un público no especializado y, a su vez, fomentar las visitas a la página web de la propia revista. Dicho bloque cumple con el segundo **objetivo secundario** del proyecto. Seguidamente, se muestra la secuencia de tareas implementadas para el desarrollo de este:
 - 1 Selección e implementación de modelos de resumen y traducción.
 - 2 Generación de publicaciones
 - 3 Almacenamiento y exportación de publicaciones.
 - 4 Diseño e implementación de interfaz interactiva.
 - **Chatbot RAG:** último bloque principal del proyecto, este se centra en el diseño e implementación de una aplicación web conversacional basada en técnicas *RAG (Retrieval Augmented Generation)*. Dicha aplicación tiene como objetivo responder de forma precisa y contextualizada las preguntas planteadas por los usuarios acerca de las enfermedades raras, basándose en el corpus previamente procesado. Este último bloque representa el **objetivo principal** del proyecto, debido a su tiempo de desarrollo, complejidad y funcionalidades implementadas. Las tareas a implementar son las siguientes:
 - 1 Indexación de documentos y embeddings
 - 2 Creación del dataset de preguntas y respuestas
 - 3 Selección y pruebas de modelo generativo
 - 4 Implementación del RAG pipeline

Cada bloque del sistema se ha realizado de forma modular, cada uno cuenta con sus propios procesos, herramientas y objetivos. Dichos bloques pueden ser analizados de forma independiente, a pesar de que exista relación entre ellos, como por ejemplo, los resultados del bloque de *Scraping* serán utilizados por los bloques posteriores.

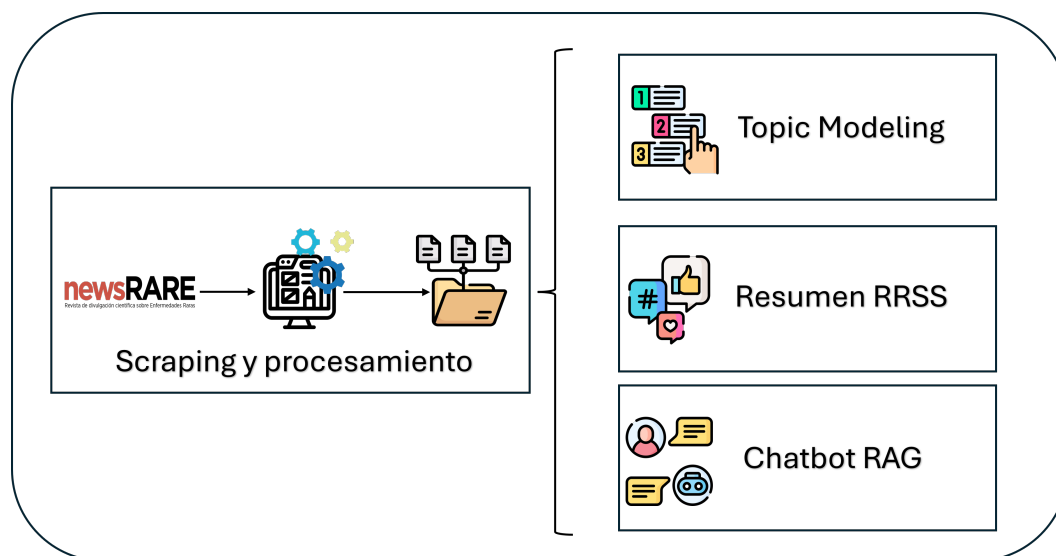


Figura 5.4: Visión conceptual del proyecto

5.2. Diseño e implementación del sistema

Tal como se ha mencionado anteriormente, se va a desarrollar en profundidad acerca de cómo es el diseño, funcionamiento e implementación de cada uno de los cuatro bloques principales comentados anteriormente en la sección 5.1. Para cualquier consulta, el código se encontrará publicado en el siguiente [repositorio](#) de *GitHub*.

5.2.1. Scraping y procesamiento

En la sección anterior se ha mencionado cómo este bloque se centra en la obtención y procesamiento de los artículos disponibles en el portal web [newsRARE](#). Para ello es fundamental conocer cómo funciona un proceso de extracción de datos a partir del código fuente de una página web, este proceso se conoce como *web scraping*.

5.2.1.1. Análisis del corpus

En primer lugar, es fundamental conocer cómo se encuentra organizada la página web de la que se va a llevar a cabo un proceso de extracción de datos o *scraping*. La siguiente figura 5.5 muestra cómo es la página principal de la revista. A simple vista, se puede observar que hay una barra horizontal con múltiples secciones, las cuales pueden contener la información deseada a extraer. Mediante un estudio de las diferentes posibilidades, se puede conocer que en la sección 'artícu-

los”se encuentran todos los artículos pertenecientes a todas las posibles secciones que posee la revista.

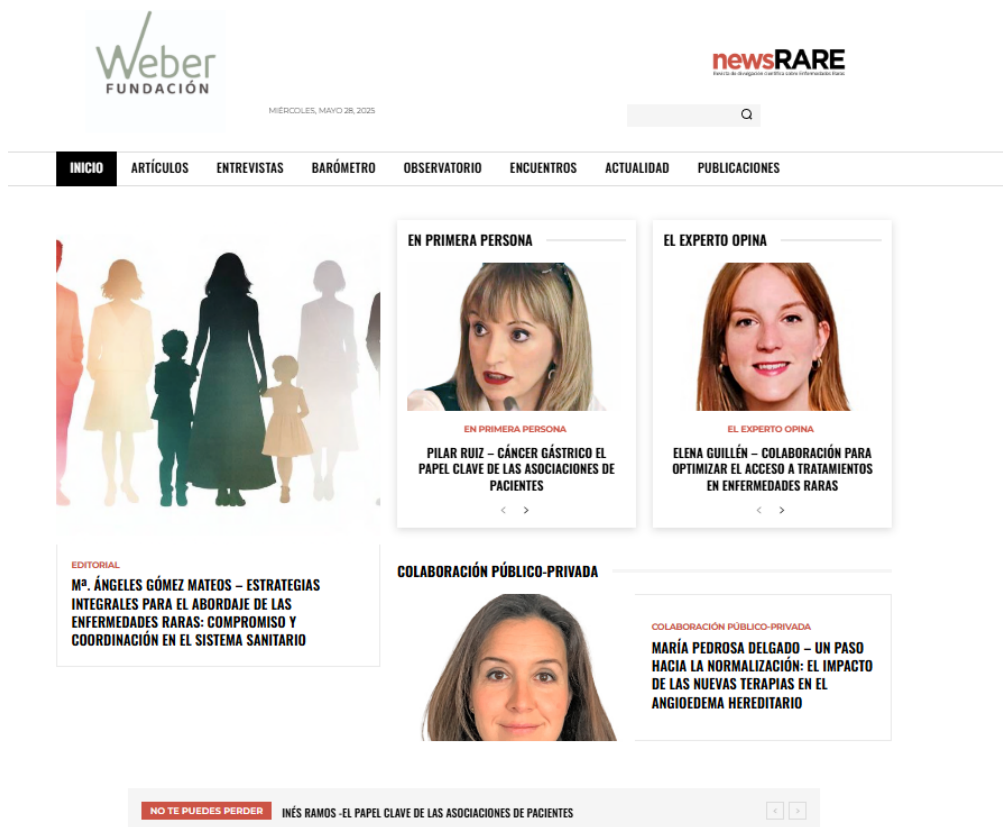


Figura 5.5: Página principal newsRARE

La figura 5.6, muestra como en la propia web se dispone de un repositorio de artículos, el cual contiene un contador de todos los artículos que hay subidos por cada sección de la revista (resaltado en el recuadro amarillo) junto a la posibilidad de navegar a los propios artículos de cada una de estas. Navegar y recopilar la información de cada uno de estos artículos de forma manual es una labor muy ineficiente y muy poco práctica, por lo que se ha de diseñar un sistema que permita la obtención de la información de cada uno de estos artículos, permitiendo recopilar de forma sencilla los nuevos artículos que se publiquen en un futuro. Actualmente, más de 200 artículos serán los que compongan el corpus de estudio y trabajo para este proyecto.

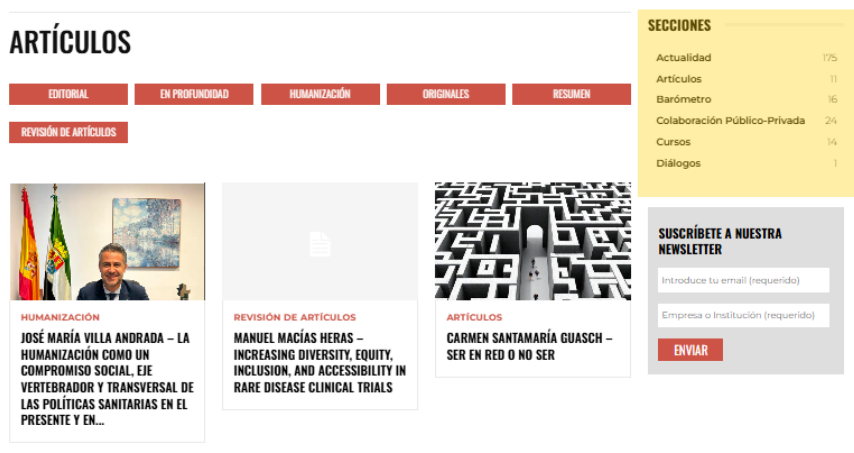


Figura 5.6: Repositorio de artículos newsRARE

5.2.1.2. Diseño del scraper

Una vez se conoce cuál va a ser la fuente de los datos, el primer paso es la obtención de las URLs correspondientes a cada sección temática. A partir de estas, se accederá a cada uno de los enlaces de los artículos individuales pertenecientes a cada una de ellas, para posteriormente extraer y almacenar la información contenida en dichos artículos. Para la creación del scraper serán necesarias dos herramientas fundamentales, estas se conocen como *Selenium* y *BeautifulSoup*.

Selenium se utiliza para simular el comportamiento que haría un usuario interactuando con el ordenador, permitiendo acceder a contenido dinámico. Esto va a ser fundamental, ya que no todos los enlaces de los artículos se cargan en cada sección. Por lo que es necesario hacer scroll manualmente para que estos carguen en la página.

Por otro lado, **BeautifulSoup** va a ser empleada para el análisis del código HTML de la página del artículo una vez esta sea cargada. Esta permite localizar con precisión los elementos deseados, como títulos, autores y el cuerpo del artículo.

A continuación, se va proceder mostrar, a modo de ejemplo, el proceso de cómo se ha llevado a cabo la obtención de las URLs de cada una de las secciones temáticas empleando BeautifulSoup.

1. **Inspección del código HTML:** a través del navegador, se puede inspeccionar la página para conocer el código fuente de los datos que queremos obtener. Pulsando F12, en el caso que se esté empleando Chrome o un navegador similar, podremos reconocer las etiquetas que se les ha otorgado a cada elemento de la página web. El código fuente de la figura 5.7, muestra que por ejemplo la clase "td-ct-item-name" contiene el nombre de la sección y "td-ct-item-no." el número de artículos que se dan para el elemento web resaltado por el cursor. A partir de esta funcionalidad se puede construir el código necesario para cumplir con el objetivo planteado.

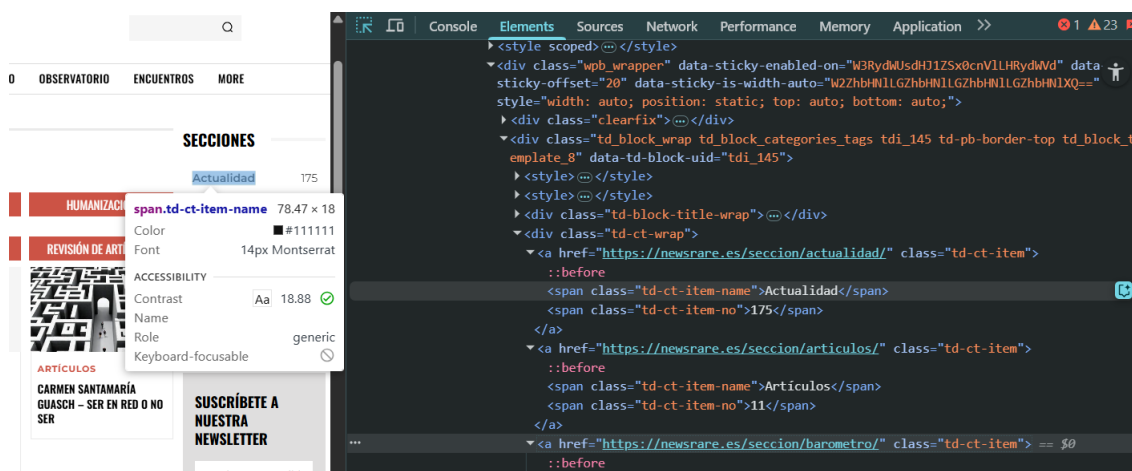


Figura 5.7: Inspección de código fuente en navegador (Chrome)

2. **Elaboración del código:** El siguiente código muestra cómo se puede acceder a las URLs de cada sección previamente inspeccionada.

```

1 import requests
2 from bs4 import BeautifulSoup
3
4 # Descargar la página
5 url = "https://newsrare.es/articulos"
6 response = requests.get(url)
7
8 # Parsear el HTML
9 soup = BeautifulSoup(response.text, "html.parser")
10
11 # Encontrar el bloque de secciones
12 section_container = soup.find("div", class_="td_block_categories_tags")
13
14 # Extraer los enlaces de las secciones
15 sections = section_container.find_all("a")
16
17 sections_urls=[]
18 nombre_seccion=[]
19 num_articulos=[]
20
21 for content in sections:
22     section_url=content["href"]
23     name=content.find("span",class_="td-ct-item-name").get_text()
24     num=content.find("span",class_="td-ct-item-no").get_text()
25
26     sections_urls.append(section_url)
27     nombre_seccion.append(name)
28     num_articulos.append(num)
29
30 df_secciones=pd.DataFrame({"Seccion":nombre_seccion,"N_Artículos":
    num_articulos,"URL":sections_urls })

```

Listing 1: Obtención de las URLs por sección con BeautifulSoup

3. **Comprobación de los resultados:** Una vez ejecutado el código, se pueden observar que los resultados obtenidos eran los esperados, tal como se muestra en el DataFrame representado en la figura.

```
[ ] df_secciones.head()
```



	Sección	NºArtículos	URL
0	Actualidad	175	https://newsrare.es/seccion/actualidad/
1	Artículos	13	https://newsrare.es/seccion/articulos/
2	Barómetro	16	https://newsrare.es/seccion/barometro/
3	Colaboración Público-Privada	24	https://newsrare.es/seccion/entrevistas/colabo...
4	Cursos	14	https://newsrare.es/seccion/encuentros/cursos/

Figura 5.8: DataFrame obtenido tras el scraping web del repositorio de artículos

El código completo del scraping se encuentra disponible en el Anexo B (Listing 7), donde se puede comprobar que, una vez se han obtenido las URLs se simula el comportamiento de un usuario para cada página de sección empleando *Selenium*. Esto permite cargar de forma dinámica todos los artículos disponibles, para posteriormente almacenarlos en formato *.json*, incluyendo el título y *URL* de cada artículo. Por último, se realizará de forma individual la obtención del cuerpo siguiendo un procedimiento similar al mostrado anteriormente, empleando *BeautifulSoup*, extrayendo el cuerpo del artículo y añadiendo dicha información al *.json* original.

5.2.1.3. Limpieza, estructuración y almacenamiento de los datos obtenidos

Una vez obtenidos los artículos de cada sección, se elabora un archivo *.csv* que aglutina toda la información relevante propia de estos artículos, permitiendo persistir la información actual que se dispone de la página para su posterior análisis. Se ha optado por el formato *.csv*, ya que es directamente compatible con la librería *Pandas*, la cual permite su manipulación, actualización y gestión de los datos de una forma sencilla y eficiente.

Cabe destacar que durante la obtención de los artículos, se han realizado técnicas básicas de depuración del texto, tales como la eliminación de saltos de línea o de espacios innecesarios. No obstante, en etapas posteriores, como en el bloque de **Topic Modeling**, se llevará a cabo una limpieza mucho más profunda de los textos. Se aplicarán procedimientos tales como la eliminación de signos de puntuación y acentos, la tokenización y filtrado de palabras que no aportan información, comúnmente conocidas como *stopwords*.

5.2.1.4. Consideraciones adicionales

La obtención de los datos de la página web se ha realizado con la colaboración directa y estrecha de la revista *newsRARE*. El procesamiento de los artículos se ha realizado exclusivamente con el objetivo de aplicar las tareas enunciadas en el marco de trabajo de este proyecto y en ningún caso se ha modificado el mensaje original de dichos textos, respetando de forma plena su contenido y finalidad divulgativa.

5.2.2. Topic Modeling

Una de las tareas principales de este proyecto se centra en la identificación y estructuración de los temas principales de los artículos extraídos de la revista *newsRARE*. Para poder cumplir dicho objetivo, es necesario aplicar técnicas de procesamiento de lenguaje natural o *NLP*, las cuales no solamente facilitarán la comprensión de los datos y del contenido que se dispone, sino que también aportarán información clave que aporte luz a los resultados obtenidos de tareas posteriores. La implementación en código de cada una de las tareas que se definen a continuación se encuentra en el Anexo B (Listings 8, 9 y 10).

5.2.2.1. Preprocesamiento del texto

La labor de preprocesamiento del texto es fundamental durante la aplicación del *Topic Modeling*, esto permite la reducción del corpus, eliminando posible ruido léxico, asegurando que el contenido a analizar es más consistente y homogéneo. Los pasos que se han llevado a cabo son los siguientes:

1. **Normalización:** En primer lugar, se llevó a cabo el descarte de signos de puntuación, acentos y caracteres especiales tanto de los títulos como del cuerpo de cada artículo. Además, todo el contenido fue convertido a minúsculas con el objetivo de evitar palabras repetidas y posibles errores en el posterior análisis.
2. **Análisis exploratorio EDA:** Una vez que se dispone del texto limpio para cada artículo, se puede llevar a cabo un análisis exploratorio que permita identificar cuáles son los términos que más aparecen en un texto y comprender la distribución léxica que tiene el corpus. Mediante la técnica conocida como **term-frequency**, se lleva a cabo el registro del número de veces que aparece cada palabra en el conjunto de artículos. Esta operación se apoya en el concepto de **Bag-of-Words**, el cual es un modelo que representa cada artículo como una bolsa de palabras, donde se ignora el orden que estas poseen y solamente se contabiliza la frecuencia de aparición de cada una de ellas.



Figura 5.9: WordCloud análisis exploratorio

La anterior figura 5.9, representa una nube de palabras que permite a simple vista identificar cuáles son las palabras más repetidas en el corpus seleccionado. A simple vista, se comprueba que las palabras que más aparecen son artículos, preposiciones y conjunciones. Estas palabras, es conveniente ignorarlas en el análisis, ya que no aportan ningún conocimiento acerca de los textos que se están tratando y reciben el nombre de **stopwords**.

3. **Eliminación de stopwords:** Librerías como `nlTK`[22], disponen de colecciones de *stopwords* predefinidas que facilitan una limpieza rápida del texto. La figura 5.10 muestra el resultado de aplicar dicha librería al corpus, dando lugar a términos mucho más relacionados con el marco temático de este proyecto, con palabras tales como .^{en}fermedadesraras”, ”pacientesz ”medicamentos”. Cabe destacar que, durante el modelado de los temas, se podrán añadir al conjunto de stopwords las palabras que, en el caso de estudio, no aporten o aparezcan en todos los temas de forma recurrente.



Figura 5.10: WordCloud tras eliminación de *stopwords*

4. **Lematización:** Para homogeneizar las variantes gramaticales que disponen los términos, se ha optado por la selección de un modelo (es_core_news_sm) perteneciente a la librería `spaCy`[23], entrenado en lengua española. Este modelo permite identificar correctamente la forma base de cada palabra, *lema*. A diferencia de otros métodos como el *stemming*, que pueden generar formas carentes de significado o recortadas al buscar obtener la raíz de la palabra, el lema obtenido resulta ser una palabra real y coherente del idioma.

Término original	Stemming	Lematización
caminando	camin	caminar
corriendo	corr	correr
mejor	mejor	bueno
hablando	habl	hablar

Cuadro 5.1: Comparación entre Stemming y Lematización

5.2.2.2. Representación vectorial del texto

Para que los modelos computacionales puedan llevar a cabo la extracción de los temas de los diferentes textos, es necesario llevar a cabo un proceso de **vectorización**, donde se lleva a cabo la conversión del contenido de los artículos a representaciones numéricas. Previamente se mencionó la técnica **Bag-of-Words**, la cual fue utilizada para la identificación de los *stopwords*. Esta simplemente construye una matriz donde cada fila representa un documento y cada columna un término del vocabulario. El valor en cada celda indica la frecuencia de aparición de ese término en el documento correspondiente.

Aunque es muy eficaz y fácil de implementar para algunos casos, este modelo no tiene en cuenta el contexto que se está tratando ni la relevancia global entre los términos.

Para llevar a cabo una correcta extracción de los temas del corpus, se ha optado por implementar la técnica **TF-IDF (Term Frequency-Inverse Document Frequency)**[24]. Esta permite llevar a cabo una ponderación que permite ajustar la importancia de los términos que tiene un documento en función de lo comunes o raros que estos son en el conjunto total de documentos del corpus. Este planteamiento permite penalizar las palabras que aparecen con mucha frecuencia y destaca aquellas que son más representativas en cada documento.

La ponderación TF-IDF de un término i en un documento j se define como:

$$\text{TF-IDF}(i, j) = \text{df}(i, j) \times \log \left(\frac{N}{\text{df}(i)} \right)$$

Donde:

- $\text{tf}(i, j)$ es la frecuencia del término i en el documento j .
- $\text{df}(i)$ es el número de documentos que contienen el término t .
- N es el número total de documentos del corpus.
- El logaritmo penaliza los términos comunes y refuerza los términos más raros.

El resultado que se obtiene de aplicar TF-IDF da lugar a una matriz documento-término *sparse*, la cual puede contener celdas vacías, donde cada valor representa la importancia relativa de cada término dentro de un documento. Esta matriz sirve de entrada para los algoritmos de *Topic Modeling* desarrollados en los siguientes apartados.

Cabe destacar que un término en un documento, dependiendo de cómo se desee durante la implementación, puede referirse a una palabra única o a un **n-grama**, los cuales están compuestos por agrupaciones de palabras.

Tipo	Secuencias extraídas del texto
Texto base	Estrategias integrales para el abordaje de las enfermedades raras
Unigramas	estrategias, integrales, para, el, abordaje, de, las, enfermedades, raras
Bigramas	estrategias integrales, integrales para, para el, el abordaje, abordaje de, de las, las enfermedades, enfermedades raras
Trigramas	estrategias integrales para, integrales para el, para el abordaje, el abordaje de, abordaje de las, de las enfermedades, las enfermedades raras

Cuadro 5.2: Ejemplo de extracción de n -gramas a partir de texto

El uso de n -gramas permite la captura de unidades semánticas compuestas las cuáles podrían ser significativas para el análisis temático de los documentos. A diferencia del uso exclusivo de palabras individuales, que puede producir una pérdida del contexto, los n -gramas permiten almacenar relaciones relevantes entre las diferentes palabras, como ".enfermedades raras." ".abordaje de las". En la sección de análisis de resultados se podrá observar la relevancia de estas agrupaciones influyen en la coherencia de los temas y cuál de estas favorece una extracción más representativa.

5.2.2.3. Modelado mediante NMF

Tras la obtención de la representación vectorial del corpus mediante el uso de la técnica TF-IDF, se procede a aplicar un modelo NMF (**Non-negative Matrix Factorization**)[25]. Este modelo es una técnica ampliamente utilizada en el contexto de *Topic Modeling*.

NMF busca descomponer la matriz documento-término, la cual se obtiene tras la aplicación de TF-IDF en nuestro caso, en el producto de dos matrices no negativas.

A modo ilustrativo, el proceso de descomposición de la matriz TF-IDF mediante NMF puede representarse de la siguiente forma:

$$\underbrace{\begin{bmatrix} 0,2 & 0,1 & 0,4 \\ 0,3 & 0,0 & 0,5 \\ 0,0 & 0,6 & 0,1 \end{bmatrix}}_{\text{TF-IDF (documento-término)}} \approx \underbrace{\begin{bmatrix} 0,5 & 0,3 \\ 0,6 & 0,4 \\ 0,2 & 0,8 \end{bmatrix}}_{\text{W (documento-tema)}} \cdot \underbrace{\begin{bmatrix} 0,3 & 0,1 & 0,6 \\ 0,2 & 0,5 & 0,3 \end{bmatrix}}_{\text{H (tema-término)}}$$

Cada fila de la matriz W representa la distribución de temas en un documento, mientras que cada fila de la matriz H indica la importancia de cada término dentro de un tema.

NMF no busca la factorización exacta de la matriz original, sino encontrar, mediante un problema de optimización, qué matrices W y H permiten minimizar el error entre la matriz original y la matriz reconstruida. Mediante la librería **scikit-learn**, la implementación de NMF resulta mucho más sencilla y exige al usuario de realizar dichas tareas de optimización.

Como detalle matemático, la implementación que realiza dicha librería de NMF, se centra en la **Norma de Frobenius**, definida como la raíz cuadrada de la suma de los cuadrados de las diferencias entre los elementos de la matriz original y los de la matriz reconstruida. La optimización se lleva a cabo mediante un algoritmo basado en gradiente, conocido como **Coordinate Descent**, el cual actualiza de forma alterna las matrices W y H .

La función objetivo a minimizar es la siguiente:

$$\min_{W, H \geq 0} \|V - WH\|_F^2 = \sum_{i=1}^m \sum_{j=1}^n (V_{ij} - (WH)_{ij})^2$$

Una vez implementado, el modelo se ha aplicado por separado a los títulos, textos y a la combinación de ambos. Esta estrategia permite explorar el corpus documental de forma más amplia, analizando tanto el contenido completo de los artículos como la información sintetizada en sus títulos. Además, facilita la identificación de posibles relaciones temáticas consistentes entre los títulos y los textos, lo que puede aportar valor a nivel semántico y estructural en el análisis posterior.

A continuación, la siguiente figura muestra los resultados obtenidos para los primeros documentos del corpus. Estos corresponden a las palabras con mayor peso dentro de la matriz H . Gracias a esto, cada documento puede relacionarse a un conjunto de palabras clave que describen su temática principal, facilitando así su clasificación y posterior análisis.

5.2.2.4. Visualización y asignación de temas/tópicos

Por último, ya obtenidos los tópicos a los que pertenece cada artículo, es necesario buscar la forma de representación que permita extraer las conclusiones más adecuadas respecto a cómo de representativos son los temas extraídos para cada documento, y si estos pueden agruparse en grupos temáticos bien diferenciados.

Para ello, es necesario reducir la dimensionalidad de los resultados obtenidos. Dependiendo del número k de temas que se deseen extraer por ejecución, se va a obtener una matriz documento-tema W , donde cada documento es un vector de k dimensiones. Estos vectores no pueden visualizarse de forma directa, por lo que es necesario aplicar técnicas de reducción de dimensionalidad como **t-SNE (Stochastic Neighbour Embedding)**[26]. Esta técnica permite tomar los vectores de alta dimensión y proyectarlos en dos dimensiones, preservando las relaciones de proximidad entre documentos, es decir, los documentos que tengan temas similares estarán próximos entre sí en el plano.

Esto permite visualizar de forma gráfica y sencilla cómo se agrupan los diferentes documentos del corpus en función de los temas que se han extraído. La figura 4.2 muestra, a modo de ejemplo, algunos de los posibles grupos temáticos que se forman tras implementar *Topic Modeling*. En la sección 6.1, se tratará en detalle todos los resultados obtenidos tras la implementación del código, el cual se encuentra referenciado en el Anexo B 7.2, donde también se encuentran todos los bloques implementados desde el preprocesamiento hasta la visualización final de los tópicos.

5.2.3. Automatización de resúmenes

El segundo de los objetivos secundarios marcados en la fase de planificación de este proyecto (sección 3.2), consiste en la creación de una herramienta que permita la generación automática de textos para que estos puedan ser publicados en redes sociales, posibilitando el acercamiento a conocimientos enfocados en las enfermedades raras a un público no especializado.

La accesibilidad es uno de los pilares principales sobre los que se basa este proyecto. Por este motivo, las herramientas seleccionadas buscan replicar los resultados que pueden obtenerse mediante soluciones comerciales de pago, como las APIs de modelos generativos tipo *Claude* o *ChatGPT*, las cuales, gracias a su potencia y versatilidad, ofrecen muy buenos resultados con gran facilidad de uso.

Ante esta situación, se ha optado por un enfoque alternativo que permita obtener resultados satisfactorios empleando únicamente herramientas de acceso libre, como las disponibles a partir de la plataforma. **HuggingFace**.

5.2.3.1. Selección e implementación de modelos de resumen y traducción.

Tal como se ha mencionado anteriormente, se va a hacer uso de la plataforma *HuggingFace* para la selección de los modelos que permitan llevar a cabo la generación automática de los resúmenes del corpus obtenido. Se han planteado dos enfoques diferentes:

1. Enfoque empleando **traducción intermedia**: los modelos *NLP* especializados en resumir artículos médicos presentan limitaciones a la hora de trabajar con textos en castellano. Por lo tanto, es necesario hacer uso de modelos de traducción que permitan hacer la conversión de castellano-inglés y viceversa para poder hacer un uso correcto del modelo de resumen. La siguiente figura 5.11, muestra la secuencia que se da a cabo empleando los diferentes modelos para llevar a cabo el resumen de los artículos.

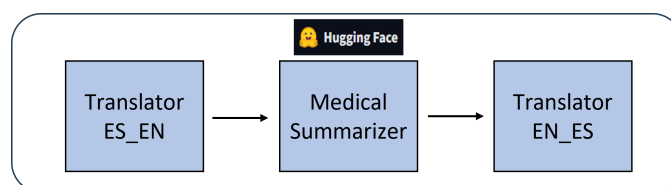


Figura 5.11: Secuencia lógica para la traducción/resumen de artículos

Los modelos que se han seleccionado son los siguientes:

- **Traducción**: los modelos seleccionados para llevar a cabo la traducción de los artículos del español al inglés y viceversa son el modelo [Helsinki-NLP/opus-mt-es-en](#) y [Helsinki-NLP/opus-mt-en-es](#) respectivamente. Están basados en arquitecturas *Transformer* y presentan una calidad de traducción y eficiencia más que suficiente para esta aplicación a realizar.
 - **Resumen**: para la generación de los resúmenes, se ha optado por el modelo [Falcon-sai/medical.summarization](#), especializado en textos biomédicos. Como se ha mencionado anteriormente, este modelo requiere entradas en inglés y proporciona modelos condensados que capturan los elementos más relevantes del texto, además de términos médicos clave tales como medicamentos, síntomas y tecnicismos que aportan gran valor al resumen.
2. Enfoque directo en **castellano**: como alternativa, se ha probado el empleo de modelo de resumen que puede trabajar directamente con contenidos en castellano. Esta estrategia presenta la ventaja de simplificar el *pipeline* a costa del uso de un modelo con menos capacidad a priori. Se utiliza el modelo [csebutnlp/mT5_multilingual_XLSum](#), una variante multilingüe basada en *T5* (modelo diseñado por Google) que ha sido entrenada con artículos de prensa en más de 40 idiomas, incluido el español. Permite realizar la tarea de resumen sin necesidad de preprocesamiento lingüístico adicional.

5.2.3.2. Generación de publicaciones

Una vez se dispone de un *pipeline* que permite el resumen de los artículos, es importante formatear de la manera deseada dichos resúmenes en función de la publicación que se desee generar. Cada red social presenta diferentes límites de caracteres, por lo que es importante determinar el tipo de plataforma sobre la que se va a publicar para adecuar el resumen al formato específico que se desee tratar.

Por ejemplo, la plataforma **X**, popularmente conocida anteriormente como Twitter, limita las publicaciones a 280 caracteres, lo cual es una diferencia relativamente considerable con otros medios populares tales como **LinkedIn** (3000 caracteres) o **Instagram** (2,200 caracteres).

La estructura del post sí será común a todos los formatos, cada publicación debe de contener el **título** del artículo, seguido del **resumen** para posteriormente indicar la **url** del artículo para que los usuarios puedan acceder de forma sencilla a la noticia publicada en el portal de **newsRARE**.

5.2.3.3. Diseño de la herramienta

Una vez se conoce el *pipeline necesario* desde la generación del resumen a la estructuración del post, se ha diseñado e implementado una herramienta empleando el framework de **Gradio**, el cual permite generar de forma automática un post adaptado en tamaño a la red social deseada, a partir de una *URL*. De esta forma, la lógica empleada en el Scraping (Sección 5.2.1) para la obtención de artículos, se ve reimplantada, permitiendo de una forma mucho más sencilla llevar a cabo el resumen automático de los artículos deseados sin tener que operar de forma manual con el contenido de estos.

Gradio permite mediante código Python, la creación de interfaces visuales muy sencillas a través de bloques, también conocidos como widgets, los cuales solo hace falta instanciarlos mediante una serie de parámetros para alcanzar la funcionalidad deseada. Esta herramienta es muy potente y presenta mucho potencial a la hora de desplegar demos funcionales atractivas de una forma sencilla y rápida. El siguiente bloque de código (Listing 2) muestra cómo en pocas líneas se puede llevar a cabo la creación de la interfaz empleada en esta herramienta.

```
1
2 # Gradio
3
4 with gr.Blocks() as demo:
5
6     gr.Markdown(""" # RAREpost \n ## Resume automáticamente los artículos
7                     deseados para crear una publicación en la RRSS seleccionada """)
8
9     url_input = gr.Textbox(label=" URL del artículo", placeholder="https://...")
10
11     red_social = gr.Dropdown(choices=["Twitter (X)", "LinkedIn", "Instagram"],
12                             label=" Red social destino", value="Twitter (X)")
13
14 with gr.Row():
```

```
13
14     texto_original = gr.Textbox(label=" Texto extraído", lines=10)
15
16     resumen_generado = gr.Textbox(label=" Resumen generado", lines=6)
17
18     publicacion = gr.Textbox(label="Publicación generada", lines=6)
19
20     btn_generar = gr.Button("Generar publicación")
21
22     btn_generar.click(fn=pipeline_resumen, inputs=[url_input, red_social],
23                       outputs=[texto_original, resumen_generado, publicacion])
24
25 demo.launch()
```

Listing 2: Implementación de interfaz empleando Gradio

The screenshot shows a web application titled "RAREpost" with a dark theme. The main heading reads "Resume automáticamente los artículos deseados para crear una publicación en la RRSS seleccionada". The interface includes several input fields: "URL del artículo" with a placeholder "https://...", "Red social destino" with a dropdown menu currently set to "LinkedIn", "Texto extraído" (a large empty text area), "Resumen generado" (a smaller empty text area), and "Publicación generada" (a large empty text area). At the bottom, there is a prominent button labeled "Generar publicación".

Figura 5.12: Interfaz herramienta de resumen automatizado

La figura 5.12, muestra una sencilla interfaz gráfica que permite al usuario introducir el enlace al artículo de la revista *newsRARE*. Este artículo será procesado y su información será empleada para

la elaboración secuencial del resumen, seguido de la generación automática del post. Además, se le permite al usuario observar el artículo obtenido al completo, la salida generada por el modelo de resumen para posteriormente en la parte inferior encontrar un recuadro con el post generado listo para postear en la red social seleccionada.

Un ejemplo visual de los resultados que se obtienen tras el uso de la herramienta correspondería a la figura 5.13, la cual se muestra a continuación:

The screenshot displays the RAREpost web application interface. At the top, it reads 'Resume automáticamente los artículos deseados para crear una publicación en la RRSS seleccionada'. Below this, there are input fields for the article URL and the target social media platform (set to LinkedIn). The main area is split into two columns: 'Texto extraído' on the left and 'Resumen generado' on the right. The 'Texto extraído' section contains a paragraph about innovative financing models and lists several examples, such as 'Acuerdos de pago por resultados y riesgo compartido' and 'Acuerdos de techo máximo de gasto'. The 'Resumen generado' section contains a concise summary of the article's main points. At the bottom, a 'Publicación generada' section shows a preview of the social media post, including a title, a short text snippet, and the original URL. A 'Generar publicación' button is located at the very bottom of the interface.

Figura 5.13: Ejemplo de resultados obtenidos mediante herramienta de resumen automatizado

En la siguiente sección 6.3, se llevará a cabo la evaluación y estudio de los resultados obtenidos a partir de la herramienta.

5.2.4. Chatbot RAG (Retrieval Augmented Generation)

El bloque final del proyecto se enfoca en el diseño e implementación de un sistema *Q/A* (Question/Answer) especializado en enfermedades raras. Este sistema se elaborará a partir de las técnicas RAG, previamente mencionadas en la sección 2.2. A diferencia de los modelos generativos clásicos, el enfoque RAG plantea un enfoque híbrido que combina la capacidad de estos modelos con la recuperación de información relevante, lo cual garantiza una mayor trazabilidad y fidelidad de las respuestas generadas por el sistema.

La arquitectura RAG implementada junto a la herramienta de interacción usuario-chatbot, se muestra en la siguiente imagen.

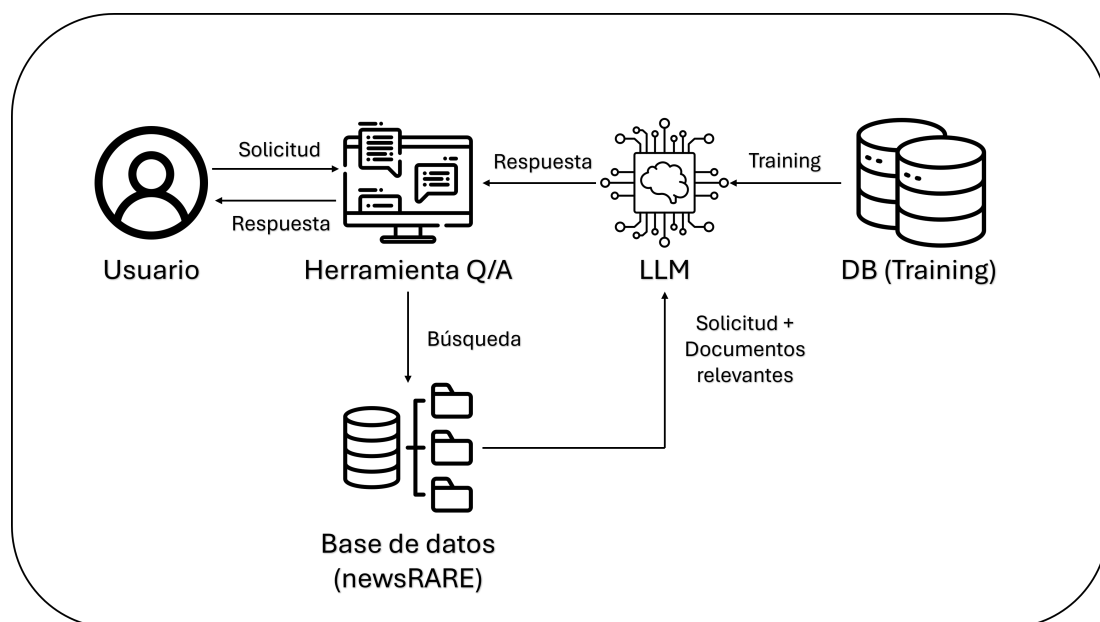


Figura 5.14: Arquitectura RAG implementada

A continuación, se describe el papel de cada uno de los elementos que forman parte de esta arquitectura:

- **Herramienta Q/A:** interfaz applicativa, que se realizará empleando **Gradio**, a partir de la cual el usuario va a interactuar con el sistema. Este sistema deberá de ser capaz de responder las preguntas realizadas siempre que estas se encuentren relacionadas con el estudio de las enfermedades raras. En caso de no darse la información o la pregunta se encuentre fuera del marco del proyecto, el sistema deberá hacerle saber al usuario esta información.
- **Base de datos propia:** mediante el uso de técnicas de **embeddings** y la herramienta **ChromaDB**, se lleva a cabo la construcción de una base de datos sobre la que se llevará a cabo la recuperación de los documentos más relevantes a la consulta que realice el usuario en cada

momento.

- **LLM pre-entrenado:** se ha optado por el uso de modelos pre-entrenados con bases de datos compuestas por millones de parámetros y documentos, tales como *BioMedLM* o *Meta-Llama-3* para llevar a cabo una respuesta coherente ante las preguntas realizadas por el usuario, basándose en la información recuperada de la base de datos de *embeddings*.

Langchain ha sido seleccionada como la librería encargada de cohesionar los distintos elementos de la arquitectura previamente mostrada. Esta herramienta facilita la construcción de agentes conversacionales basados en **LLMs**, permitiendo integrar de forma sencilla tanto los modelos de generación, como las **bases vectoriales** y las plantillas personalizadas para realizar consultas al modelo (**prompting**).

5.2.4.1. Creación de base de datos vectorial (**embeddings**)

Para poder almacenar la información de cada uno de los artículos recopilados, es necesario crear una base de datos que recopile y facilite la recuperación de estos contenidos de forma eficiente ante las preguntas del usuario. El proceso de creación de la base de datos se divide en tres tareas diferenciadas:

1. **Segmentación (Chunking) del contenido:** cada uno de los artículos será segmentado en fragmentos de longitud fija, los cuales presentarán un solapamiento entre bloques, para evitar pérdidas de contexto entre fragmentos consecutivos.
2. **Generación de embeddings:** los segmentos o chunks obtenidos en el paso anterior son convertidos a vectores normalizados (**embeddings**). Se emplearán diferentes modelos del tipo **embedding transformer**, como [BAAI/bge-m3](#) y [all-MiniLM-L6-v2](#) para su posterior comparación y evaluación.
3. **Almacenamiento en base de datos vectorial:** se procede a almacenar de forma persistente los embeddings y sus metadatos (título, url) en la base de datos vectorial **ChromaDB**, la cual puede funcionar de forma local, es ligera y permite búsquedas en tiempo real.

5.2.4.2. Creación del dataset de preguntas y respuestas para la evaluación

Es necesaria la creación de un dataset de preguntas y respuestas que nos permita establecer un criterio objetivo para la evaluación de las respuestas generadas por el sistema. Para la elaboración de estas preguntas y respuestas se empleará también una IA generativa debido a la gran extensión del corpus documental.

El modelo de IA seleccionado para la evaluación **no** será empleado para la generación de las respuestas dentro del sistema. Esto se debe a que, para facilitar la generación del dataset de preguntas, se ha optado por un modelo de altas capacidades accesible mediante suscripción **GPT-4o**. Modelos tan avanzados no pueden ser implementados en el sistema *RAG*, ya que comprometerían uno de los objetivos clave del proyecto: la accesibilidad. Además, resulta inviable desplegar un modelo de tales características en espacios gratuitos como Google Colab y HuggingFace Spaces, ya que la limitación de recursos impide una ejecución eficiente.

Para la obtención de preguntas se ha optado por extraer la información de todas las revistas publicadas por newsRARE hasta la fecha (25 revistas publicadas) y extraer un total de 20 preguntas por revista, lo cual da lugar a un total de 500 preguntas.

Volúmenes	Pregunta y Respuesta
<p>NUM.1 - FEBRERO 2016</p>	<p>1. ¿Qué es una enfermedad rara según la Unión Europea?</p> <p>Una enfermedad rara es aquella que afecta a menos de 5 de cada 10.000 personas en la Unión Europea, lo que equivale a menos de 245.000 europeos.</p>
	<p>2. ¿Cuántas enfermedades raras se han identificado hasta la fecha?</p> <p>Se han descubierto más de 6.000 enfermedades raras distintas.</p>
	<p>3. ¿Qué porcentaje de enfermedades raras tiene origen genético?</p> <p>El 80 % de las enfermedades raras tienen origen genético.</p>
	<p>4. ¿Cuál es el promedio de tiempo que transcurre entre la aparición de los primeros síntomas y el diagnóstico de una enfermedad rara?</p> <p>El promedio de tiempo estimado es de 5 años, aunque en uno de cada cinco casos puede tardar 10 años o más.</p>
<p>NUM.2 - NOVIEMBRE 2016</p>	<p>1. ¿Qué son los registros poblacionales y cuál es su finalidad?</p> <p>Los registros poblacionales son sistemas de información organizados que recogen todos los casos de una enfermedad en un ámbito geográfico definido. Su finalidad es estudiar la historia natural de la enfermedad, generar indicadores epidemiológicos y planificar recursos sanitarios.</p>
	<p>2. ¿Existe un sistema universal de codificación de enfermedades raras en España?</p> <p>No existe un sistema universal de codificación para enfermedades raras en España. Sin embargo, iniciativas como Orphanet y la CIE-10 han mejorado la visibilidad de estas enfermedades en los sistemas sanitarios.</p>
	<p>3. ¿Qué avances se han logrado en Navarra en el área de enfermedades raras?</p> <p>Navarra ha desarrollado el Registro Poblacional de Enfermedades Raras de Navarra (RERNA) y ha participado en el proyecto SpainRDR, además de contar con grupos de investigación en genética y enfermedades neuromusculares.</p>
	<p>4. ¿Qué papel juegan los pacientes en los registros estatales de enfermedades raras?</p> <p>Los pacientes aportan información sobre su calidad de vida y experiencias diarias, lo que ayuda a mejorar los registros y la investigación. También participan activamente en la recopilación de datos y en la toma de decisiones.</p>

Cuadro 5.3: Muestra de dataset de preguntas para los volúmenes publicados por newsRARE

5.2.4.3. Selección del modelo generativo

El modelo generativo *LLM (Large Language Model)* a emplear de *HuggingFace* debe de ser capaz de elaborar respuestas coherentes y precisas a partir de los documentos recuperados de la base de datos vectorial. Los diferentes modelos que se han seleccionado se basan en el equilibrio entre coste computacional y rendimiento. Como se ha mencionado anteriormente, los recursos computacionales de la herramienta se encuentran limitados, pero esto no implica que la solución planteada no trate de ofrecer la mejor calidad de servicio y experiencia de usuario posible.

Modelo	NºParámetros/ Tamaño	Dominio
microsoft/biogpt	355M (1.6 GB)	Biomédico
microsoft/BioGPT-Large-PubMedQA	1.5B (6.38 GB)	Biomédico
google/flan-t5-large	780M (3 GB)	General
google/flan-t5-xl	3B (12 GB)	General

Cuadro 5.4: Comparativa de modelos generativos evaluados

Dicha selección mostrada en la tabla 5.4, responde a la necesidad de llevar a cabo la comparación entre modelos de tamaños similares pero entrenados empleando enfoques diferentes, es decir, se pueden agrupar los modelos en dos categorías o dominios:

- **Modelos de propósito general:** los modelos *Flan-T5* y *Flan-T5-large* han sido entrenados con temas variados y están adaptados a responder instrucciones generales. Los modelos T5 son muy populares entre la comunidad debido a su flexibilidad y posible *fine-tuning* hacia diferentes aplicaciones.
- **Modelos especializados en el dominio biomédico:** en cambio, los modelos *BioGPT*, se han entrenado específicamente en literatura biomédica, por lo que es de gran relevancia si dichos modelos se adaptan de manera satisfactoria al dominio de las enfermedades raras. Es de importancia destacar que, para su funcionamiento, es necesario tener en cuenta que estos modelos no son multilingües, por lo que se deberán de implementar los mismos bloques de traducción que los empleados en la sección 5.2.3.

En la sección 6.4, se contemplará el rendimiento obtenido de los diferentes modelos para posteriormente seleccionar cuál es el más adecuado en términos de coherencia, fiabilidad y rapidez en la generación de las respuestas.

5.2.4.4. Implementación del RAG pipeline

Por último, es necesario llevar a cabo la implementación e interconexión de todos los elementos que figuran en el diagrama 5.14. A continuación, se describe el proceso lógico de implementación. Tal como se ha realizado en apartados anteriores, el código de esta implementación se encuentra adjuntado en el Anexo B (Listing 13).

En primer lugar, se debe cargar el modelo de embeddings, el cual será el mismo que se haya empleado en la indexación del corpus (en el Anexo B se emplea **BAAI/bge-m3**). Dicho modelo permitirá representar de forma vectorial las preguntas que realice el usuario y estas serán comparadas con los vectores almacenados en la base de datos previamente persistida en **ChromaDB**. A partir de esta base de datos se genera un recuperador de documentos **retriever**, el cual devuelve los fragmentos más relevantes en cada consulta realizada.

En segundo lugar, se debe definir una plantilla **prompt**, esta plantilla estructura las respuestas generadas, instruyendo al modelo para que se base únicamente en la información extraída del contexto recuperado. A continuación se carga el modelo generativo en cuestión a través de *Hugging Face*, el cual se conecta a LangChain estableciendo un *HuggingFacePipeline*. La cadena RAG **RetrievalQA** será la encargada de coordinar todos estos elementos, desde la recuperación de la información hasta la generación final de la respuesta.

Por último, se implementa una interfaz interactiva mediante *Gradio*, a través del componente **ChatInterface**, se permite la creación de una interfaz interactiva que permite al usuario dialogar con el sistema en lenguaje natural y recibir respuestas fundamentadas en el corpus documental junto a los fragmentos reales recuperados por el sistema para que el propio usuario pueda corroborar la respuesta generada por el modelo. Adicionalmente, el usuario puede crear múltiples chats, permitiendo una personalización de las diferentes consultas que el usuario desee plantear a la herramienta. Las siguientes figuras 5.15 y 5.16 muestran la interfaz generada:

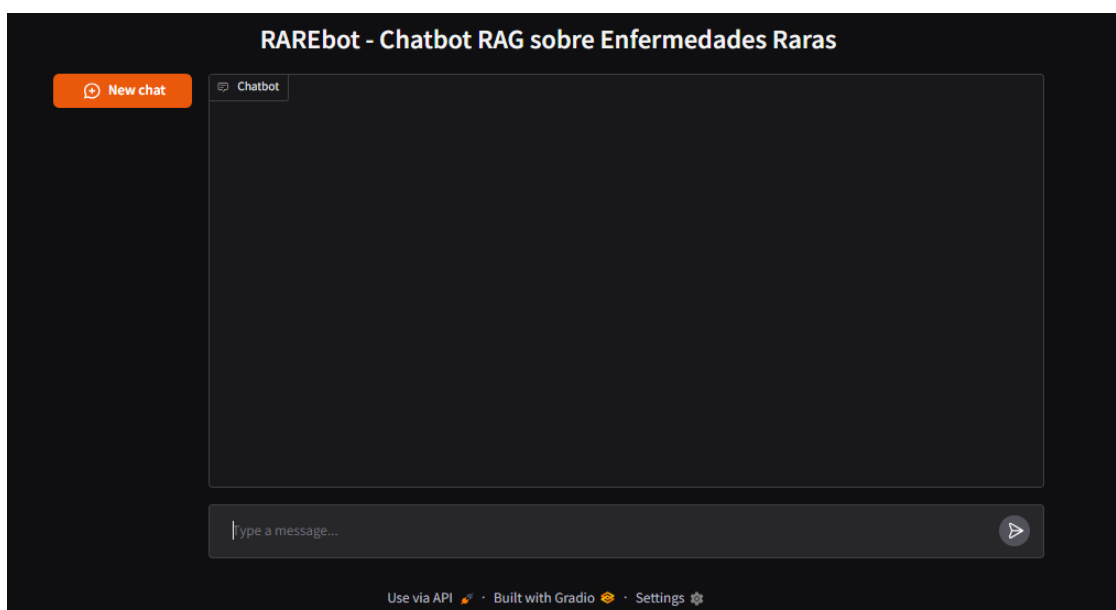


Figura 5.15: Interfaz del chatbot RAG (Recién iniciada la aplicación)



Figura 5.16: Interfaz del chatbot RAG (Ejemplo de uso)

6. Análisis de los resultados

Una vez se ha tratado en detalle el diseño e implementación de todos los componentes que conforman el proyecto, es momento de llevar a cabo la evaluación de los resultados obtenidos por cada uno de estos bloques principales. Este proceso de evaluación es fundamental para poder llevar a cabo la detección de los puntos fuertes del sistema e identificar los posibles aspectos que puedan ser mejorados en desarrollos futuros.

6.1. Análisis de los temas extraídos

Para llevar a cabo la evaluación de los temas extraídos, lo primero que se debe de plantear son los criterios de evaluación que se van a tener en cuenta a la hora de determinar. Se ha optado por un análisis en tres niveles: **cuantitativo**, **cualitativo** y **visual**.

6.1.1. Evaluación cuantitativa

La primera fase de la evaluación tiene como finalidad medir la calidad semántica de los temas extraídos a partir de métricas numéricas. Las métricas seleccionadas [27] son ampliamente utilizadas en el ámbito del *Topic Modeling*, se ha implementado a partir de la librería **gensim** y son las siguientes:

- **Coherencia C_v (métrica principal)**: métrica basada en la coocurrencia de los términos dentro de un contexto (ventana deslizante). A partir de los textos de los artículos tokenizados, se genera un diccionario con todos los términos y su frecuencia. Una vez que se extraen los temas, se emplean medidas de segmentación y de similitud de coseno para obtener la métrica. C_v proporciona valores entre 0 y 1, un valor más alto indica que las palabras clave de un tópico suelen aparecer juntas en el corpus y, por lo tanto, este tópico es semánticamente coherente.
- **Coherencia C_{umass} (métrica de control)**: métrica de control basada en el cálculo de la probabilidad condicional de que las palabras extraídas de un tópico aparezcan juntas, esta métrica permite detectar tópicos con palabras incoherentes e irrelevantes. Tiende a ser negativo y cuanto más próximo este se encuentre de 0, menos incoherentes serán las palabras de los tópicos extraídos.

Los modelos fueron entrenados con distintas configuraciones de número de tópicos ($k = 5, 7, 10, 12$ y 15), tamaños de n-gramas ($n = 1, 2, 3$) y sobre los títulos, artículos y la mezcla de ambos. Los resultados de coherencia C_v y C_{umass} se representan en los siguientes gráficos de líneas:

Evolución de Coherencia C_v por tipo de texto y tamaño de n-grama

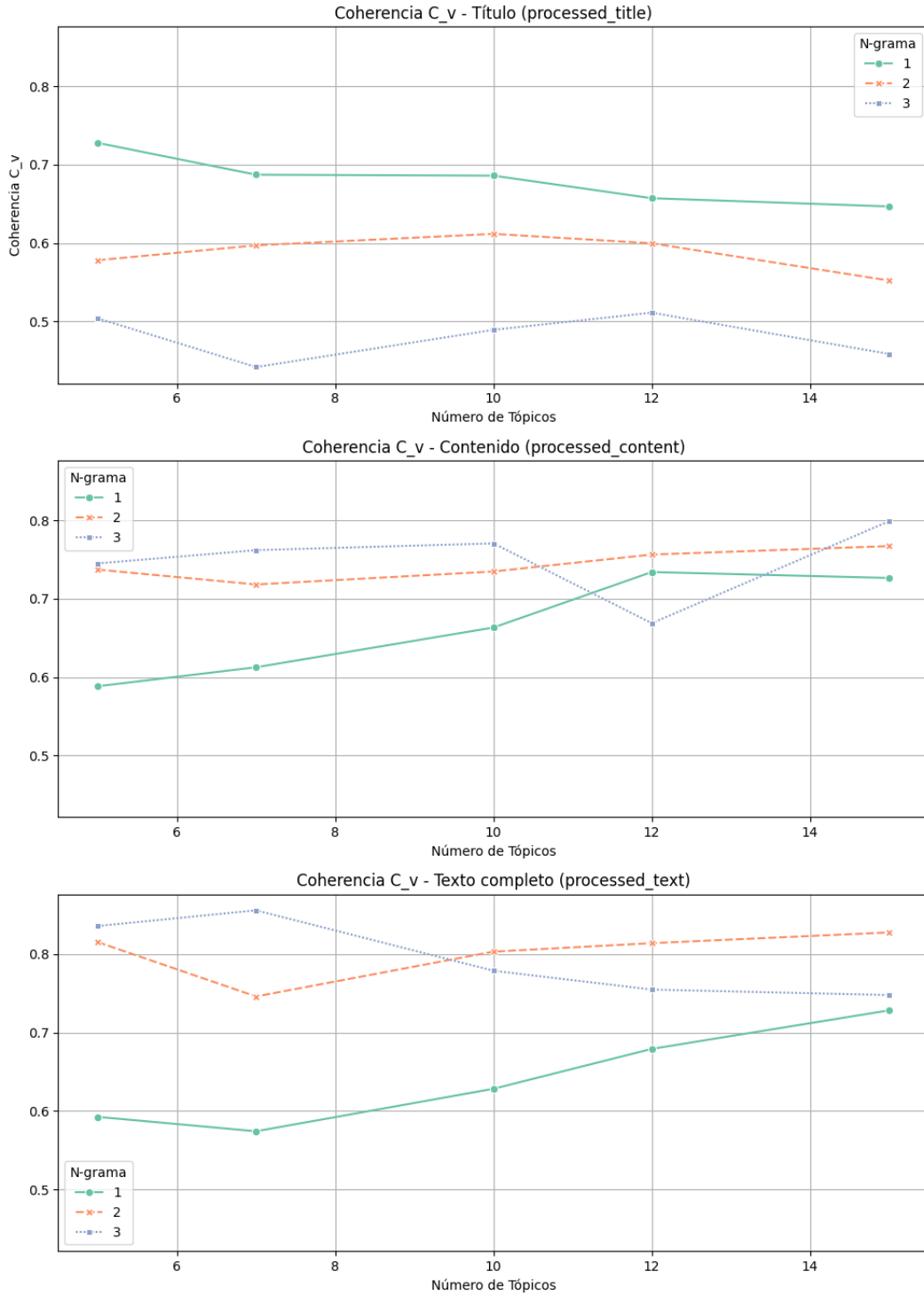


Figura 6.17: Coherencia C_v en función del nº de topics y tamaño de n-grama

Evaluación de Coherencia C_umass por tipo de texto y tamaño de n-grama

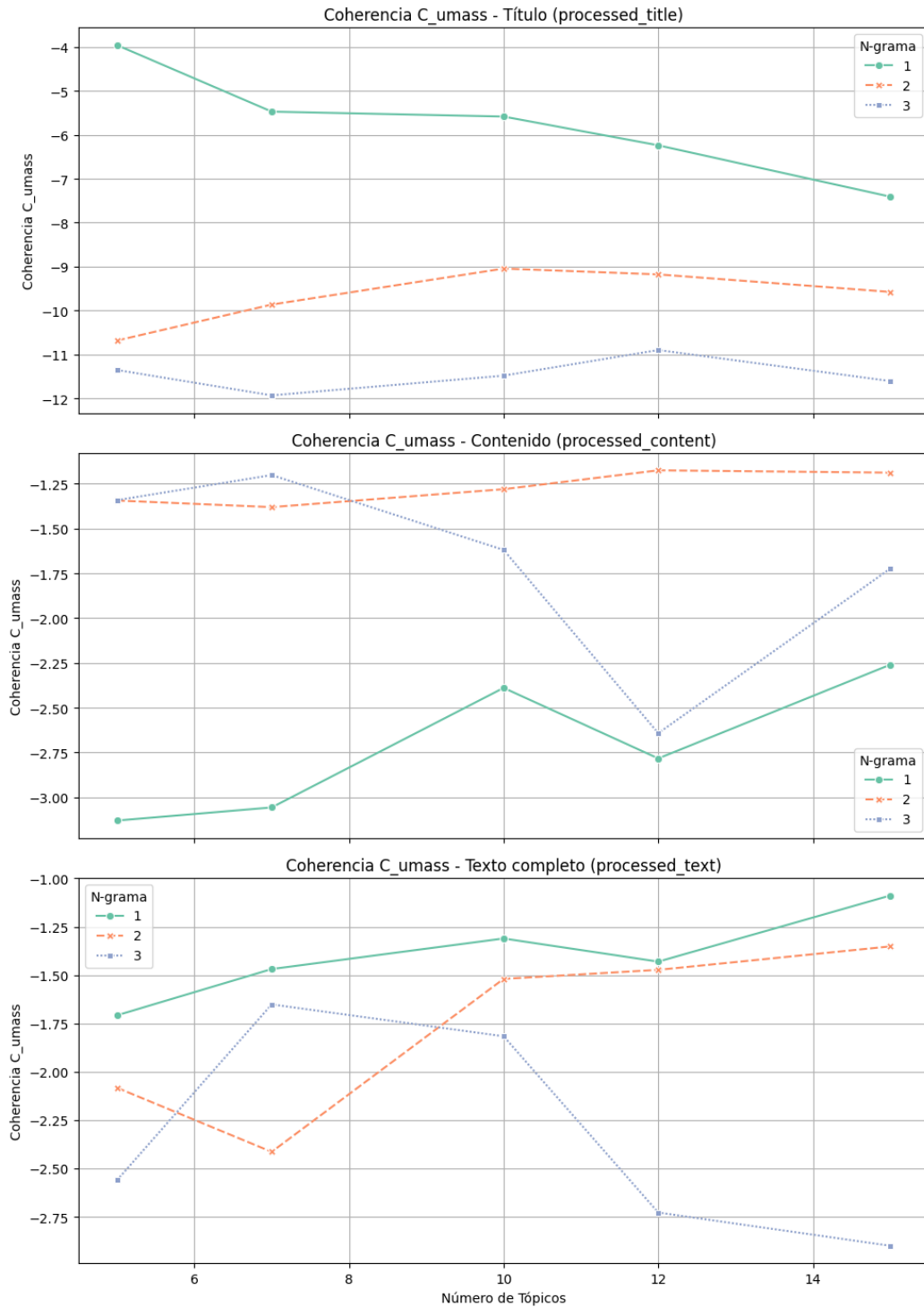


Figura 6.18: Coherencia C_umass en función del nº de topics y tamaño de n-grama

Respecto a la figura 6.17, se puede observar lo siguiente a partir de las métricas obtenidas en función de los temas extraídos de los diferentes textos:

- **Título:** mejor rendimiento de los **unigramas**, mientras que, los bigramas y trigramas tienen una coherencia menor, posiblemente causada por la escasa longitud de los textos, lo cual limita la aparición repetida de expresiones compuestas.
- **Cuerpo del artículo:** la mejor coherencia se alcanza empleando **trigramas**. No obstante, cabe destacar que el uso de **bigramas** parece presentar una tendencia más estable a medida que aumenta el número de tópicos, lo que sugiere que son menos sensibles al número de tópicos y más robustos estadísticamente..
- **Artículo completo:** entre 5 y 10 tópicos, los **trigramas** presentan una mayor coherencia semántica global, de nuevo, cabe destacar que los **bigramas** son muy consistentes (superan sistemáticamente valores de C_v por encima de 0.8), con un comportamiento estable en configuraciones con más tópicos.

Por otro lado, a partir de la figura 6.18 se pueden extraer los siguientes conocimientos:

- **Título:** los **unigramas** presentan mejores valores de C_{umass} (menos negativos), lo cual indica mayor frecuencia de coocurrencia entre palabras. En cambio, los bigramas y trigramas se alejan más de cero, lo cual refuerza la idea de que los títulos son demasiado cortos para permitir extraer tópicos con más de un término.
- **Cuerpo del artículo:** el uso de **bigramas** presenta los mejores resultados, tanto por la métrica obtenida como por la estabilidad mostrada al aumentar el número de tópicos. Los trigramas muestran alta variabilidad, lo cual sugiere problemas de que los tópicos extraídos no aparezcan en corpus más extensos.
- **Artículo completo:** empleando los artículos completos C_{umass} es mejor con los **unigramas**, seguida de los **bigramas** con valores muy similares a partir de los 10 tópicos. Los trigramas tienen la peor coherencia, lo cual refuerza que estadísticamente estos presentan una mayor rareza de aparición.

Ante las observaciones realizadas, puede concluirse que los **trigramas** pueden ser muy potentes semánticamente (C_v) pero menos frecuentes (C_{umass}). Por lo tanto, para realizar un análisis temático equilibrado entre interpretabilidad y soporte estadístico, el modelo más adecuado para la extracción de tópicos a partir del contenido completo del artículo será el que emplee **bigramas** y extraiga entre 10 y 15 tópicos, ya que parece ser el más robusto y coherente globalmente. Adicionalmente, el uso de **unigramas** parece ser fiable a la hora de llevar a cabo la extracción sobre los títulos, donde la longitud limitada del texto favorece términos individuales frente a estructuras compuestas. Sin embargo, para textos más extensos, los unigramas tienden a perder riqueza semántica, generando temas menos interpretables a nivel conceptual.

6.1.2. Evaluación cualitativa

Además de la obtención de métricas que miden la coherencia de los tópicos, es necesario evaluar dicha coherencia e interpretabilidad desde un punto de vista humano.

En primer lugar, se va a proceder a extraer los términos más representativos de cada tópico, lo cual permite identificar si los temas están centrados en aspectos significativos, presentan agrupaciones consistentes o contienen palabras inconexas, redundantes o muy genéricas que a lo mejor conviene considerar como *stopwords*

Por ejemplo, tomando el modelo de extracción de tópicos **NMF**, el cual emplea **bigramas** y del cual se extraen 15 tópicos, se procede a mostrar en la figura 6.19, la distribución de los tópicos asignados a cada uno de los artículos. La tabla 6.5 muestra las top k=6 palabras asignadas a los 4 tópicos más frecuentes que se dan en el corpus.

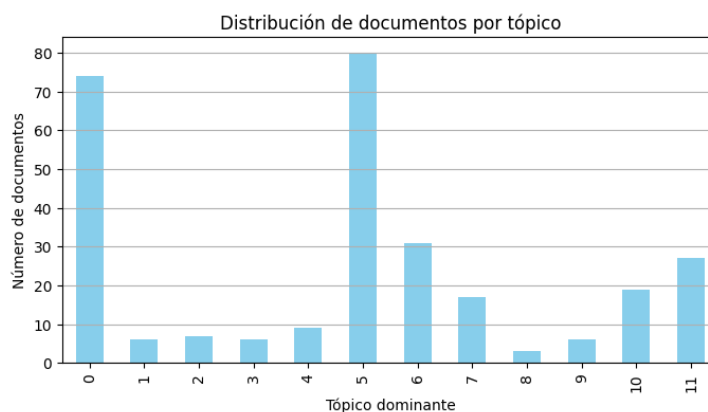


Figura 6.19: Asignación de tópicos en el corpus newsRARE

Tópico	Palabras clave
0	acceso medicamento, jazz pharmaceuticals, ensayo clínico, ser importante, acceso paciente, punto vista
5	persona enfermedad, vivir enfermedad, enfermedad raro, paciente enfermedad, persona vivir, español enfermedad
6	paciente adulto, tratamiento paciente, paroxístico nocturno, hemoglobinuria paroxístico, nocturno hpn, tratamiento adulto
11	tecnología sanitario, evaluación tecnología, evaluación medicamento, evaluación clínico, proceso evaluación, clínico conjunto

Cuadro 6.5: Palabras clave de los tópicos 0, 5, 6 y 11 extraídos mediante NMF

De los temas extraídos, se puede interpretar que el modelo de extracción de tópicos es capaz de obtener aspectos temáticos diferentes:

- **Tópico 0:** vinculación centrada en la accesibilidad a medicamentos, ensayos clínicos y el papel de empresas farmacéuticas como Jazz Pharmaceuticals en el mundo de las enfermedades raras.
- **Tópico 5:** tema vinculado a la experiencia vital de los pacientes españoles con enfermedades raras.
- **Tópico 6:** tema centrado en el abordaje de tratamientos específicos a pacientes adultos como por ejemplo la hemoglobinuria paroxística nocturna.
- **Tópico 11:** tema claramente enfocado en la evaluación tecnológica y clínica de terapias y medicamentos.

Estos temas extraídos parecen estar en consonancia con posibles temáticas técnicas específicas del dominio médico y de las enfermedades raras. Por ejemplo, la siguiente figura 6.20, muestra un fragmento de un artículo, el cual ha sido asignado al tópico 5, donde se trata acerca de la vida de los pacientes que experimentan enfermedades raras:

¿Cómo encaja la reciente campaña de concienciación de Boehringer Ingelheim sobre EERR de la piel en su enfoque general, y cuál es la relevancia de la concienciación social para mejorar el acceso y la calidad de vida de los pacientes?

MC: Las EERR de la piel son trastornos cutáneos graves y poco comunes que causan un gran impacto en la vida de las personas que las padecen. Por eso, y en línea con el compromiso de Boehringer Ingelheim con esta área, la campaña “Enfermedades raras de la piel” busca concienciar y educar sobre enfermedades dermatológicas minoritarias para dejar a un lado el estigma y la desinformación que conllevan, acercarlas a la sociedad y poner de relevancia la importancia de la investigación y el apoyo a quienes las padecen.

Figura 6.20: Fragmento [entrevista newsRARE](#)

Desde una perspectiva de utilidad, los temas que se han extraído no solo producen agrupaciones coherentes, sino que permiten también estructurar el corpus documental en nuevas dimensiones de gran utilidad para el análisis posterior o para la posible reestructuración de las noticias y artículos en el portal web.

6.2. Evaluación visual de los temas extraídos

Por último, se va a llevar a cabo un análisis visual acerca de cómo se agrupan los artículos con sus respectivas temáticas dominantes. Se ha empleado la proyección t-SNE [26], la cual permite representar en dos dimensiones las distribuciones temáticas generadas por el modelo NMF, facilitando

la interpretación visual del espacio temático.

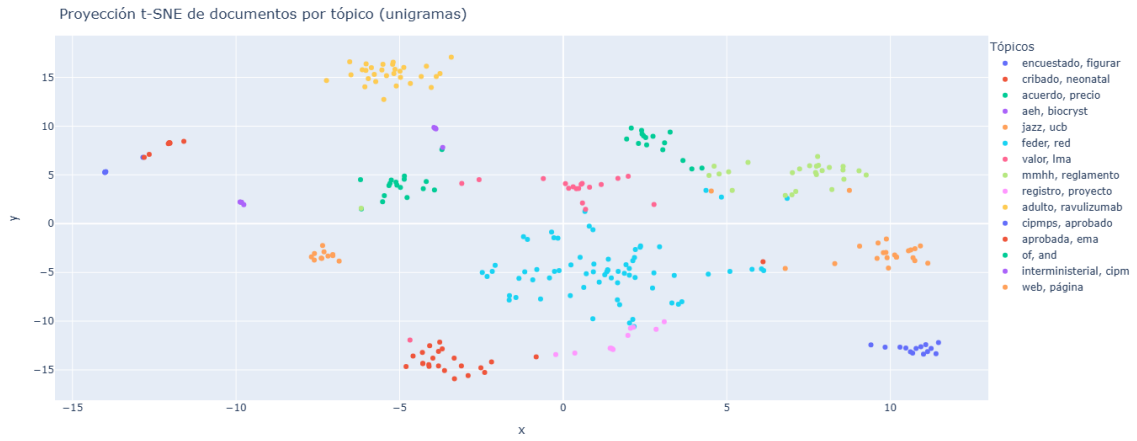


Figura 6.21: t-SNE empleando unigramas y 15 topics

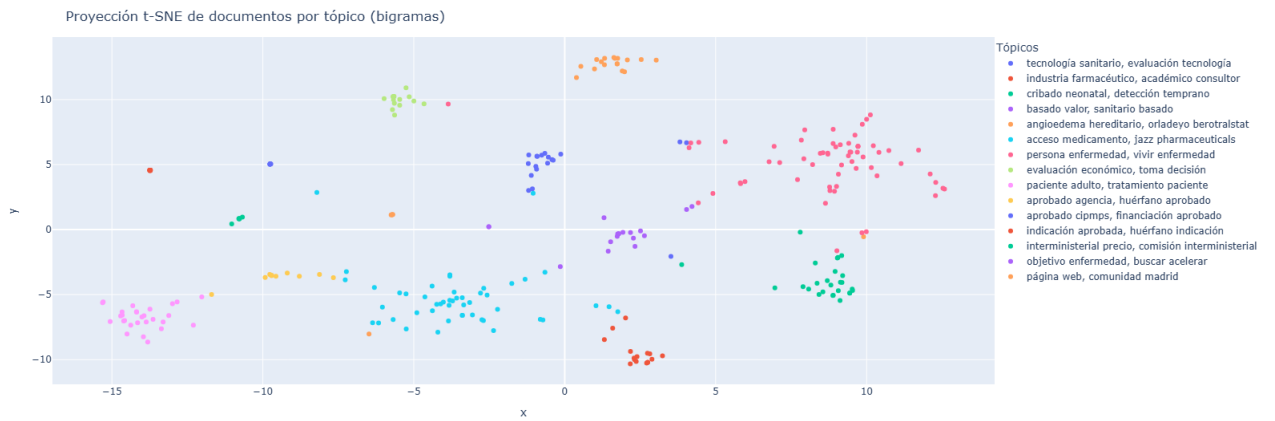


Figura 6.22: t-SNE empleando bigramas y 15 topics

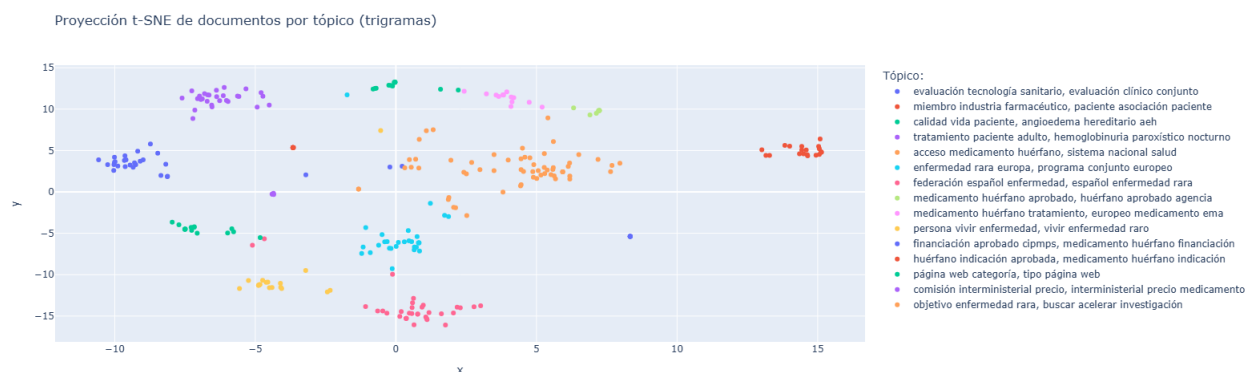


Figura 6.23: t-SNE empleando trigramas y 15 topics

Modelo de n-gramas	Temas extraídos
Trigramas	<ul style="list-style-type: none"> • Evaluación tecnología sanitaria, evaluación clínico conjunto • Acceso medicamento huérfano, sistema nacional salud • Paciente adulto, hemoglobinuria paroxística nocturno • Objetivo enfermedad rara, buscar acelerar investigación
Bigramas	<ul style="list-style-type: none"> • Cribado neonatal, detección temprana • Acceso medicamento, ensayo clínico • Evaluación económica, toma decisión • Industria farmacéutica, miembro industria
Unigramas	<ul style="list-style-type: none"> • Acuerdo, precio • Ravulizumab, adulto • Proyecto, registro • Web, página

Cuadro 6.6: Ejemplos representativos de temas extraídos por modelo de n-gramas

Las figuras anteriores reflejan cómo se agrupan los diferentes tópicos extraídos de los artículos completos en función de si se han empleado unigramas, bigramas o trigramas, y la tabla 5.2 muestra ejemplos de los temas extraídos en función de estos.

En primer lugar, empleando unigramas, se observa una mayor dispersión de los puntos en el espacio, con numerosos grupos poco diferenciados. Esto se alinea con la menor coherencia semántica (C_v) observada en la fase cuantitativa. En segundo lugar, empleando bigramas se observan agrupaciones más diferenciadas y distribuidas a lo largo del espacio. Por último, el empleo de trigramas también genera grupos densos y bien definidos, pero se da un solapamiento mayor entre grupos, produciendo que alguno de estos temas no sean tan diferenciables como en el modelo anterior.

Por lo tanto, ante las observaciones realizadas, el modelo que, tras el análisis cuantitativo, pre-

sentaba una alta coherencia semántica (C_v) y un equilibrio adecuado entre interpretabilidad y frecuencia (C_{umass}), resulta ser un modelo que produce una segmentación clara y legible.

Ante esto, es importante destacar que la extracción de temas, al ser un proceso no supervisado, requiere cierta experiencia previa en el dominio temático para realizar una correcta interpretación. Dado que no se dispone de una categorización global asociada a los artículos, no es posible contrastar de forma absoluta que la extracción de tópicos seleccionada sea la más óptima. Sin embargo, los resultados obtenidos en la evaluación tanto a nivel cualitativo, cuantitativo y visual sí demuestran que la segmentación realizada es suficientemente válida y robusta como para considerar que el **objetivo secundario (Clasificación temática de los artículos)** 3.2 propuesto en este proyecto ha sido realizado satisfactoriamente.

6.3. Evaluación resúmenes generados

A continuación, se procede a evaluar la herramienta de generación automática de resúmenes desde un punto de vista tanto cualitativo como cuantitativo, siguiendo la misma metodología planteada en la sección anterior.

6.3.1. Evaluación cuantitativa

Dado que no se dispone de un conjunto de resúmenes escritos manualmente que sirvan como referencia para la evaluación, se ha optado por el uso de la métrica **BERTScore**[28]. Esta métrica evalúa la similitud semántica entre el resumen generado y el texto original. Esta aproximación permitirá medir cómo de bien el resumen captura el mensaje del artículo sin la necesidad de disponer de una referencia previa.

A continuación, se lleva a cabo la comparativa entre los diferentes enfoques expuestos en la sección 5.2.3, los cuales abordan la generación de resúmenes desde perspectivas distintas: por un lado, mediante el uso de un modelo de resumen habilitado para textos en castellano y propósito general, y por otro lado, a través de un modelo especializado en el dominio biomédico, que requiere la traducción previa de los artículos del castellano al inglés para un funcionamiento correcto.

Se ha procedido a generar un gráfico de cajas (*boxplot*) y las tablas estadísticas que comparan el rendimiento de los dos métodos empleados.

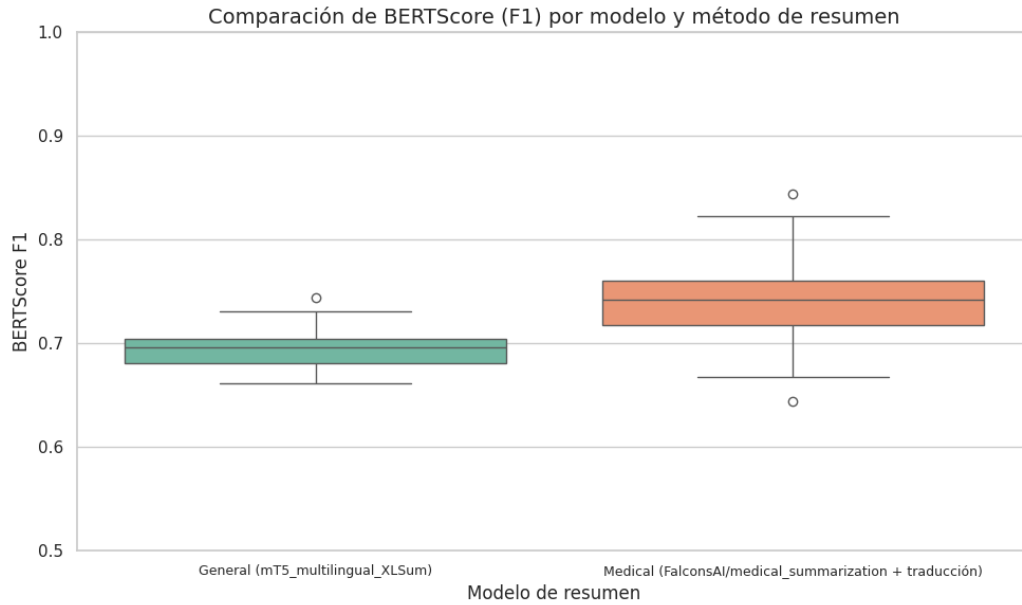


Figura 6.24: Comparativa BERTScore obtenida entre modelos (General vs Médico)

Método empleado	mean	std	min	max
General (mT5_multilingua_XLSum)	0.7401	0.0226	0.6884	0.8148
Medical (medical_summarization + traducción)	0.8287	0.0460	0.7089	0.9275

Cuadro 6.7: Métrica Bertscore Precision por modelo

Método empleado	mean	std	min	max
General (mT5_multilingua_XLSum)	0.6541	0.0191	0.6183	0.7075
Medical (medical_summarization + traducción)	0.6743	0.0407	0.5895	0.8039

Cuadro 6.8: Métrica Bertscore Recall por modelo

Método empleado	mean	std	min	max
General (mT5_multilingua_XLSum)	0.6943	0.0176	0.6611	0.7445
Medical (medical_summarization + traducción)	0.7431	0.0392	0.6437	0.8443

Cuadro 6.9: Métrica Bertscore F1 por modelo

A partir de las métricas obtenidas se pueden extraer las siguientes conclusiones:

- **Precisión:** el método **medical** obtiene métricas superiores tanto en precisión como en recall. Destacando su media de F1 de 0.7431 frente a 0.6943 y su precisión media de **0.8287** frente a **0.7401** del enfoque general.
- **Recall:** mide la cantidad de información relevante que ha sido capturada en el resumen generado. A pesar de darse una diferencia menor que en otras métricas, el enfoque **medical** supera al general, con una media de **0.6743** frente a **0.6541**.
- **F1-score:** representa el equilibrio entre la precisión y el recall. En este caso, el enfoque **medical** presenta mejores resultados, con una media F1 de **0.7431** frente a **0.6943** en el enfoque general.
- **Variabilidad (std):** la desviación típica permite analizar la estabilidad del modelo a la hora de generar resúmenes con el mismo nivel de calidad, una desviación típica mayor sugiere que el modelo es más inestable. En esta cuestión el enfoque **general** muestra un comportamiento más estable, ya que presenta una menor desviación estándar en todas las métricas.

6.3.2. Evaluación cualitativa

Debido a que las métricas automáticas no pueden capturar todos los posibles matices, se procede a llevar a cabo una evaluación humana estructurada de una serie de artículos resumidos. La rúbrica de evaluación que se ha empleado [6.10](#), trata una serie de criterios considerados clave para que un resumen sea útil, completo y correcto.

Criterio	Pregunta a evaluar
Cobertura	¿El resumen incluye los puntos clave del artículo original?
Precisión	¿La información es veraz y carece de alucinaciones?
Coherencia textual	¿El texto generado es lógico y fluido?
Adecuación lingüística	¿Se hace uso de terminología técnica de forma correcta?.

Cuadro 6.10: Rúbrica para la evaluación cualitativa de resúmenes automáticos

Una vez conocidos los criterios de evaluación, se procede a evaluar la calidad de los resúmenes en función del enfoque planteado. Procedamos a analizar el siguiente [artículo](#) de newsRARE.



Figura 6.25: Noticia empleada para el análisis cualitativo de los resúmenes generados

A continuación, se muestran fragmentos de los resúmenes generados:

Resumen generado (modelo general)

Una encuesta sobre la innovación terapéutica en el ámbito de las áreas de tratamiento de la Universidad Autónoma de Madrid (EERR) fue publicada este miércoles en la redacción de newsARE. La Encuesta Económica para el Desarrollo y la Evaluación de Recursos Alternativos (EERR) de la Universidad Autónoma de Madrid (UNAM) señala que el mayor número de personas que participaron en estas estrategias tienen un nivel de conocimiento. En una encuesta realizada por el Instituto de Enfermedades Respiratorias y Terapias Revolucionarias de Reino Unido (EERR, por sus siglas en inglés) en Estados Unidos, solo un 8 % reportaron niveles muy bajos o muy altos de conocimiento...

Resumen generado (modelo médico)

El objetivo de este barómetro fue evaluar el conocimiento, la percepción y las expectativas relacionadas con los avances terapéuticos en el campo del EERR. Se abordan cuestiones clave sobre estrategias terapéuticas innovadoras, la importancia de la innovación o la posición de España en la investigación, así como ejemplos de enfoques terapéuticos innovadores. A través de esta encuesta, se busca comprender mejor cómo se percibe y valora la innovación terapéutica en el tratamiento del erRR, e identificar tendencias y áreas prioritarias para la futura investigación y desarrollo terapéuticos. El objetivo de este barómetro es analizar el nivel de conocimiento de los encuestados sobre los avances en la innovación terapéutica, así como su percepción de estas estrategias y su importancia. materiales y métodos la mayoría de los encuestados (n = 38; 62 %) innovación terapéutica en el campo de la ER ...

Siguiendo la evaluación a partir de la rúbrica planteada, donde cada criterio es evaluado del 1 al 5, se obtienen estos resultados:

Criterio	General	Medical	Comentario
Cobertura	1	5	El enfoque Medical recoge casi todos los temas tratados: objetivo del barómetro, distribución de participantes, nivel de conocimiento, enfoques terapéuticos destacados, percepción sobre la innovación, impacto en calidad de vida, etc. El resumen General menciona instituciones ficticias, inventa encuestas y no aborda ninguna de las secciones del artículo real.
Precisión	1	4	El modelo Medical comete algunas inexactitudes (números cruzados, confusiones como “ERRR” o fragmentos redundantes), pero su contenido es realista y está basado en el texto. El General inventa datos completamente irreales (como encuestas de la Universidad Autónoma de Madrid sobre enfermedades inventadas).
Coherencia textual	2	4	El Medical es extenso pero bien estructurado, con lógica narrativa. Presenta redundancias y errores menores en la forma. El General tiene frases inconexas, cambios abruptos de tema y estructuras fragmentadas.
Adecuación lingüística	2	4	El Medical usa términos técnicos válidos: “terapia génica”, “medicina de precisión”, “nanomedicina”, aunque hay algunas incoherencias con siglas como “ERRR”. El General mezcla instituciones, introduce errores de contexto y presenta una terminología muy confusa.
Suma Total	6/20	17/20	El enfoque Medical es el más adecuado.

Cuadro 6.11: Evaluación cualitativa de los modelos de resumen automático

Este proceso demuestra que la evaluación cualitativa ante métodos no supervisados es fundamental, ya que según las métricas cuantitativas (**BERTScore**), el modelo **General** parecía que aportaría un rendimiento aceptable. Revisando los resultados, se muestra una realidad diferente para textos extensos, el modelo tiende a generar contenido alucinado, mezclar conceptos y desviarse significativamente del mensaje original del artículo. Por otro lado, el modelo **Medical** presenta un mayor rendimiento, logrando condensar los principales bloques temáticos del artículo. No obstante, este también es víctima de caer en problemas tales como las redundancias, errar en terminología y fallos en la escritura.

Además, se confirma que el esfuerzo adicional de emplear recursos y esfuerzo computacional adicional para aplicar los procesos de traducción se ve compensado claramente, dado que los resultados obtenidos, si bien no son los óptimos, sí resultan más satisfactorios y, por lo tanto, se puede considerar que el **objetivo secundario (Resumen automático de contenidos)** se ha completado. Es muy importante destacar que es **imprescindible** realizar una revisión posterior por parte de un experto. Esto se debe a que los resúmenes no solo tienen que ser comprensibles, sino que también deben ser rigurosos desde el punto de vista médico y terminológico, ya que su objetivo se centra en su difusión por las redes.

6.4. Evaluación chatbot

En esta sección, se va a realizar la evaluación detallada del chatbot RAG implementado, el **bloque principal** del proyecto. Se debe estudiar si la herramienta generada cumple con cada uno de los objetivos planteados en el capítulo 3.2. A diferencia de las secciones anteriores, el estudio se va a realizar atendiendo a los aspectos de **veracidad, robustez, velocidad y escalabilidad** previamente definidos. Cada uno de estos aspectos será examinado desde una perspectiva tanto cuantitativa como cualitativa.

6.4.1. Veracidad

Esta sección corresponde al objetivo fundamental del bloque, este se centra en que la información proporcionada por la herramienta sea cierta y esté basada en la información recopilada a través de los artículos. Será necesario analizar tanto de forma cuantitativa como cualitativa la calidad de las respuestas obtenidas con cada uno de los modelos seleccionados para así conocer cuál es el más adecuado a implementar en este caso.

6.4.1.1. Análisis cuantitativo

Las métricas que se van a evaluar son las siguientes:

- **Similitud coseno:** calcula la similitud entre los vectores de *embedding* de las respuestas esperadas y generadas.
- **ROUGE-L:** mide la superposición literal de palabras y frases entre la respuesta generada y la esperada.
- **BERTScore-F1:** anteriormente ya utilizada, métrica basada en calcular la similitud semántica de las palabras generadas con las esperadas.

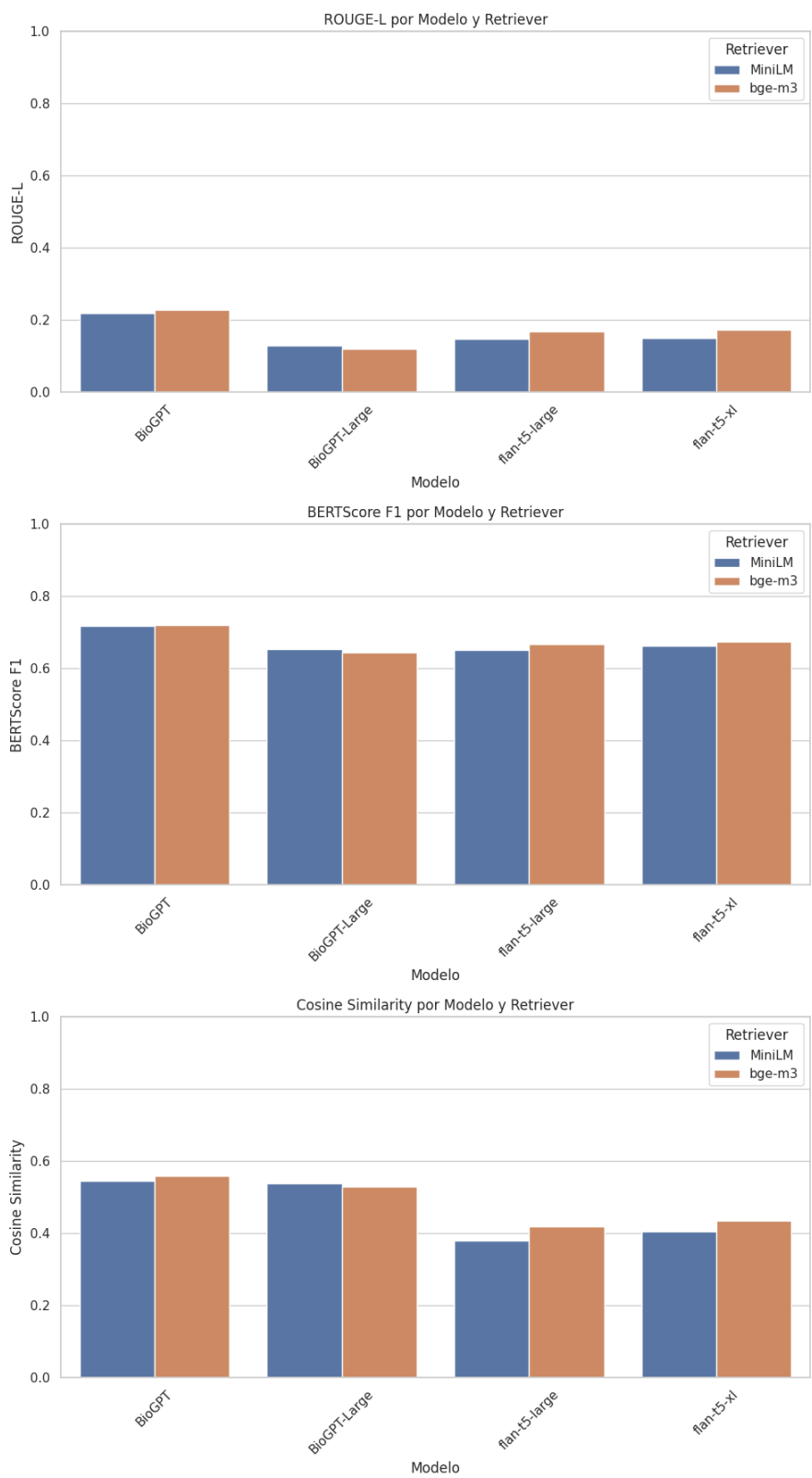


Figura 6.26: Métricas obtenidas evaluación por LLM y retriever

A partir de los *subplots* de la figura 6.26 anterior, se pueden extraer las siguientes observaciones de los siguientes puntos clave:

- **Modelo de embeddings (retriever)**: parece que la base de datos creada a partir del modelo de *embeddings* MiniLM resulta ser ligeramente inferior en todas las métricas evaluadas. En cambio, **bge-m3** proporciona contextos más ricos y relevantes, lo que se traduce en respuestas de mayor calidad por parte de los modelos generativos. Por tanto, **bge-m3** se posiciona como el modelo de recuperación más adecuado para el sistema implementado.
- **Similitud coseno**: tras aplicar el proceso de traducción y retraducción, el modelo **BioGPT** alcanza el valor más alto en esta métrica. Esto indica que sus respuestas, a pesar de no coincidir literalmente con las esperadas, presentan una gran coherencia global en cuanto al contenido temático. Los modelos `flan-t5`, por el contrario, ofrecen un rendimiento inferior en esta métrica, lo que sugiere una menor capacidad para mantener una alineación semántica amplia con la respuesta esperada.
- **ROUGE-L**: esta métrica, orientada a evaluar el solapamiento literal entre cadenas de palabras, refleja una mejora significativa en **BioGPT** respecto a evaluaciones anteriores. Tras introducir el proceso de traducción, este modelo supera incluso a las variantes `flan-t5`, lo cual indica que sus respuestas se ajustan mucho más al lenguaje y estructura de las respuestas esperadas.
- **BERTScore-F1**: al evaluar la similitud semántica contextualizada, **BioGPT** destaca nuevamente como el modelo con mejor rendimiento. Los modelos `flan-t5` mantienen un rendimiento consistente, aunque inferior al de `BioGPT` y `BioGPT-Large`.

Tras la obtención de estos resultados, parece que el modelo **BioGPT** es el más adecuado para el contexto de este proyecto. Al contrario de lo esperado, este modelo supera en rendimiento a su contraparte más potente, la cual ha sido entrenada específicamente para la respuesta a preguntas de carácter biomédico mediante el dataset *PubMedQA*. Si bien esta versión más avanzada está optimizada para ofrecer respuestas precisas en dominios médicos, su alto grado de especialización parece limitar su capacidad de adaptación a formatos más abiertos como el empleado en este sistema. Al indicarle en el **prompt** que emplee la información propia de la página **newsRARE**, este presenta menos flexibilidad que **BioGPT** y, por lo tanto, presenta una menor interpretabilidad de los datos proporcionados, lo que repercute teniendo un rendimiento inferior.

6.4.1.2. Análisis cualitativo

Ahora debe de realizarse un análisis humano de las respuestas generadas por los diferentes modelos. A continuación, se muestran una tablas comparativas (6.13, 6.14) entre las respuestas de los modelos ante una serie de preguntas pertenecientes al dataset, junto a las métricas obtenidas en la tabla 6.12 para su contraste con la evaluación cuantitativa anterior.

Pregunta	Modelo	ROUGE-L	BERTScore-F1	Cosine Similarity
Holoclar®	BioGPT	0.336	0.780	0.854
	BioGPT-Large	0.132	0.694	0.727
	Flan-T5-Large	0.134	0.697	0.266
	Flan-T5-XL	0.222	0.742	0.350
SpainRDR	BioGPT	0.243	0.678	0.448
	BioGPT-Large	0.115	0.644	0.531
	Flan-T5-Large	0.282	0.687	0.441
	Flan-T5-XL	0.167	0.660	0.481

Cuadro 6.12: Comparación de métricas de evaluación por modelo para las dos preguntas seleccionadas del conjunto de preguntas.

Pregunta	¿Qué beneficios ofrece el medicamento Holoclar® para enfermedades de la córnea?
Respuesta Esperada	Holoclar® utiliza células madre para reparar el epitelio corneal dañado, reduciendo el riesgo de rechazo y mejorando las posibilidades de éxito en trasplantes corneales.
BioGPT	Holoclar ® es un tratamiento de células madre que se puede utilizar para reemplazar el epitelio corneal dañado.
BioGPT-Large	Holoclar ® es un nuevo tratamiento para las enfermedades corneales que se ha desarrollado en la última década. Es el primer medicamento de ingeniería tisular aprobado para su comercialización en la Unión Europea. Se espera que reduzca la necesidad de trasplantes corneales y el riesgo de rechazo. Sin embargo, todavía no está claro si será rentable. Holoclar ® es un tratamiento complejo que requiere un estrecho seguimiento para asegurar que se utiliza adecuadamente. Todavía no está claro si será rentable. La empresa que fabrica Holoclar ® también está desarrollando otros tratamientos para enfermedades corneales, y todavía no está claro si estos tratamientos estarán disponibles en el mismo marco temporal que Holoclar ®. Por lo tanto, todavía no está claro si Holoclar ® será el primer medicamento de ingeniería tisular aprobado para su comercialización en la Unión Europea. Es probable que otros tratamientos sigan su ejemplo.
flan-t5-large	No se encontró suficiente información en la base de datos para responder a esta pregunta.
flan-t5-xl	Reduce el riesgo de rechazo con respecto al riesgo asociado al trasplante del tejido de un donante y no requiere cirugía en el otro ojo del paciente

Cuadro 6.13: Comparación de respuestas generadas por los modelos para la pregunta sobre el medicamento Holoclar®

Pregunta	¿Qué es el proyecto SpainRDR y cuál es su objetivo?
Respuesta Esperada	SpainRDR busca desarrollar un Registro Nacional de Enfermedades Raras en España, armonizando los registros autonómicos y promoviendo la investigación y la planificación sanitaria.
BioGPT	El proyecto SpainRDR es un proyecto de la Unión Europea (UE), que es miembro de la Unión Europea (UE).
BioGPT-Large	El proyecto se basa en el desarrollo de un registro nacional de enfermedades raras, con el objetivo de apoyar a los Estados miembros de la UE en el desarrollo de registros de pacientes comparables y compatibles en campos clínicos de cierta importancia (por ejemplo, enfermedades crónicas, enfermedades raras, etc.). El proyecto se basa en dos líneas estratégicas: registros de pacientes orientados a los resultados y registros basados en la población dirigidos a la investigación epidemiológica, social y sanitaria y a la planificación de la salud.
flan-t5-large	El Registro Nacional de Enfermedades Raras en España, a partir de dos líneas estratégicas diferentes: los registros de pacientes orientados a resultados y los registros de base poblacional dirigidos a la investigación epidemiológica, socio-sanitaria y la planificación en salud.
flan-t5-xl	Su objetivo es desarrollar el Registro Nacional de Enfermedades Raras en España, a partir de dos líneas estratégicas diferentes: los registros de pacientes orientados a resultados y los registros de base poblacional dirigidos a la investigación epidemiológica, socio-sanitaria y la planificación en salud. SpainRDR es un proyecto financiado por el Instituto de Salud Carlos III para los años 2012-2014, al amparo del Consorcio Internacional de Investigación de Enfermedades Raras (IRDiRC, por sus siglas en inglés).

Cuadro 6.14: Comparación de respuestas generadas por los modelos para la pregunta sobre el proyecto SpainRDR

Atendiendo a los resultados, se procede a realizar el análisis cualitativo:

Pregunta Holocar

En esta pregunta el modelo **bioGPT** ofrece una respuesta concisa pero acertada, donde no reproduce todos los detalles, pero sí transmite la esencia de la respuesta esperada, haciendo referencia al uso de células madre para la regeneración del tejido. Esto justifica que se hayan obtenido buenas métricas tanto de *Cosine Similarity* (0.854) y un BERTScore-F1 de 0.780. Por otro lado, se encuentra la respuesta de **BioGPT-Large**, la cual genera una respuesta mucho más extensa, con redundancias las cuales generan cierta confusión.

Esto repercute sobre su rendimiento, obteniendo métricas inferiores a las de BioGPT. **Flan-T5-Large** simplemente indica que no encuentra suficiente información, lo cual evidencia una falta de capacidad para aprovechar el contexto proporcionado. Este resultado pone de manifiesto que el valor de BERTScore por sí solo (0.694) no es suficiente para evaluar la calidad de las respuestas, siendo imprescindible complementar su análisis con otras métricas como ROUGE-L (0.134) y similitud de coseno (0.266), las cuales son muy inferiores a una respuesta mucho más completa como la obtenida a partir de **BioGPT**.

Finalmente, **Flan-T5-XL** proporciona una respuesta parcialmente correcta al mencionar aspectos como el menor riesgo de rechazo y evitar cirugía en el otro ojo, pero omite elementos clave como el uso de células madre, lo que se refleja en unas métricas más discretas, especialmente en *ROUGE-L* y similitud de coseno, pero estas son ligeramente superiores a las de **Flan-T5-Large**.

Pregunta SpainRDR

Esta pregunta evidencia la correlación entre las métricas y la calidad de la respuesta. El modelo **Flan-T5-XL** es el que ofrece una respuesta más completa, incluyendo tanto el objetivo principal del proyecto (la creación del Registro Nacional de Enfermedades Raras en España) como detalles relevantes sobre su financiación, el marco temporal y la vinculación con el Consorcio Internacional de Investigación de Enfermedades Raras. Esta riqueza informativa se traduce en las métricas más altas de todos los modelos para esta pregunta, con un *BERTScore-F1* de 0.660 y una *Cosine Similarity* de 0.481, a pesar de que el *ROUGE-L* (0.167) es relativamente bajo, probablemente debido a diferencias de redacción respecto a la respuesta esperada.

Flan-T5-Large también ofrece una respuesta alineada con el contenido esperado, enfocándose correctamente en las dos líneas estratégicas del proyecto, aunque omite detalles específicos como la financiación y los años de implementación. Sus métricas, especialmente *ROUGE-L* (0.282) y *Cosine Similarity* (0.441), son ligeramente inferiores a las de su versión más potente, pero aún reflejan una respuesta relevante.

En contraste, la respuesta de **BioGPT-Large**, aunque extensa y formal, se enfoca en generalidades sobre registros de pacientes en la UE y menciona de forma genérica algunas estrategias del proyecto. Sin embargo, no nombra explícitamente a SpainRDR ni se centra directamente en el caso español, lo que compromete su precisión. Esto se refleja en métricas bajas como *ROUGE-L* (0.115) y *Cosine Similarity* (0.531), aunque su *BERTScore-F1* de 0.644 indica cierta similitud semántica con la respuesta esperada. Finalmente, **BioGPT** genera una respuesta errónea y sin sentido, repitiendo frases sin aportar información relevante sobre el proyecto. La ausencia de contenido útil se manifiesta en métricas bajas como *ROUGE-L* (0.243) y una *Cosine Similarity* de apenas 0.448. Este ejemplo pone de relieve cómo una respuesta gramaticalmente correcta puede ser completamente inútil desde el punto de vista informativo.

Por lo tanto, ante los resultados observados, en términos de **veracidad** el modelo más competente es **BioGPT**, ya que presenta en agregado las mejores métricas obtenidas, y tal como se ha observado, la correlación entre las métricas y la calidad de la respuesta es cierta. Por lo tanto, se puede afirmar que este modelo es el más adecuado para ofrecer respuestas veraces, fieles al contenido recuperado y sin alucinaciones.

6.4.2. Robustez

En este apartado se va a estudiar si el sistema se resiste a responder preguntas que se encuentren fuera del dominio del proyecto. Para ello, tal como se mencionó durante la implementación, la librería **LangChain** permite introducir estos límites al modelo a través de la generación de una plantilla de *prompt*, la cual se encarga de realizar la consulta al modelo *LLM*.

Se va a llevar a cabo una comparación entre tres plantillas diferentes, estas tratan de comunicarle al modelo que solo debe responder a la pregunta cuando se disponga de la información necesaria para formular una respuesta.

■ Plantilla nº 1:

```
1 "plantilla_1": (  
2     "Responde únicamente con la información contenida en el contexto  
   proporcionado.\n"  
3     "Si no puedes responder con el contexto, simplemente indica: "  
4     "'No se encontró suficiente información en la base de datos para  
   responder a esta pregunta.'\n\n"  
5     "Contexto:\n{context}\n\nPregunta: {question}\n\nRespuesta:"  
6 ),
```

Listing 3: Código implementación plantilla 1

■ Plantilla nº 2:

```
1 "plantilla_2": (  
2     "Solo puedes responder si el contexto proporciona la información  
   explícita. "  
3     "Si no es así, responde únicamente: 'Fuera del dominio.'\n\n"  
4     "Contexto:\n{context}\n\nPregunta: {question}\n\nRespuesta:"  
5 ),
```

Listing 4: Código implementación plantilla 2

■ Plantilla nº 3:

```
1 "plantilla_3": (  
2     Tu respuesta debe limitarse exclusivamente al contenido del  
   contexto. "  
3     "Si no tienes suficiente información, responde: 'No disponible en  
   el contexto.'\n\n"  
4     "Contexto:\n{context}\n\nPregunta: {question}\n\nRespuesta:"  
5 ),
```

Listing 5: Código implementación plantilla 3

Además de la comparativa entre plantillas, se va a evaluar el rendimiento de los modelos de *embedding* (detallados en la sección 5.2.4.1), encargados de la recuperación de los documentos de

la base de datos propia para proporcionárselos al modelo *LLM*, en los que se basará para formular la respuesta. Para esta evaluación, se va a proceder a emplear el modelo **google/flan-t5-large**, debido a que presenta el menor coste computacional, lo cual es de gran utilidad para este tipo de evaluaciones comparativas entre diferentes modelos.

Empleando un dataset de 30 preguntas fuera de dominio, tales como "¿Quién ganó el Mundial de fútbol en 2022?", "¿Qué es la relatividad?." "¿Quién fue Beethoven?", se han obtenido los siguientes resultados:

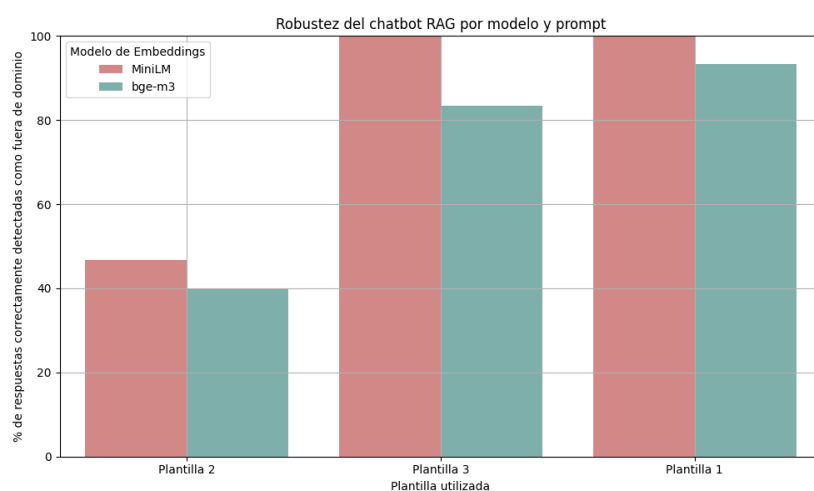


Figura 6.27: Comparativa de robustez del chatbot RAG (% de preguntas correctamente detectadas)

Observando los resultados obtenidos, se puede concluir que tanto la plantilla 1 como la plantilla 2 son más robustas (ambas superando el 80 % de preguntas correctamente identificadas) a la hora de no responder a preguntas que se encuentren fuera del dominio registrado en la base de datos vectorial.

En lo que a modelos de embedding encargados de la comparación vectorial, se puede observar que **MiniLM** es completamente efectivo (100 % de acierto) a la hora de identificar las preguntas realizadas. Por otro lado, **bg3-m3** se comporta con mayor acierto con la **plantilla 1**, por lo que esta será la plantilla seleccionada para realizar las consultas para los dos posibles modelos de *embedding* contemplados en este proyecto.

Cabe destacar lo importante y delicado que es el proceso de *prompt engineering*. Es fundamental conocer cuál es la mejor forma de realizar consultas a un modelo, por lo que es necesario tener una gran experiencia con los modelos *LLM* y realizar múltiples pruebas para poder maximizar la capacidad del sistema y obtener los mejores resultados posibles.

6.4.3. Velocidad

A continuación, se va a llevar a cabo la evaluación de la rapidez que muestra cada uno de los modelos *LLMs* evaluados y sus posibles modelos de *embedding*, los cuales son los encargados de recuperar los documentos necesarios para llevar a cabo la resolución de las consultas por parte del usuario.

Se han sometido los modelos a una evaluación sobre una misma muestra de 10 preguntas del dataset previamente creado y se han obtenido los siguientes resultados:

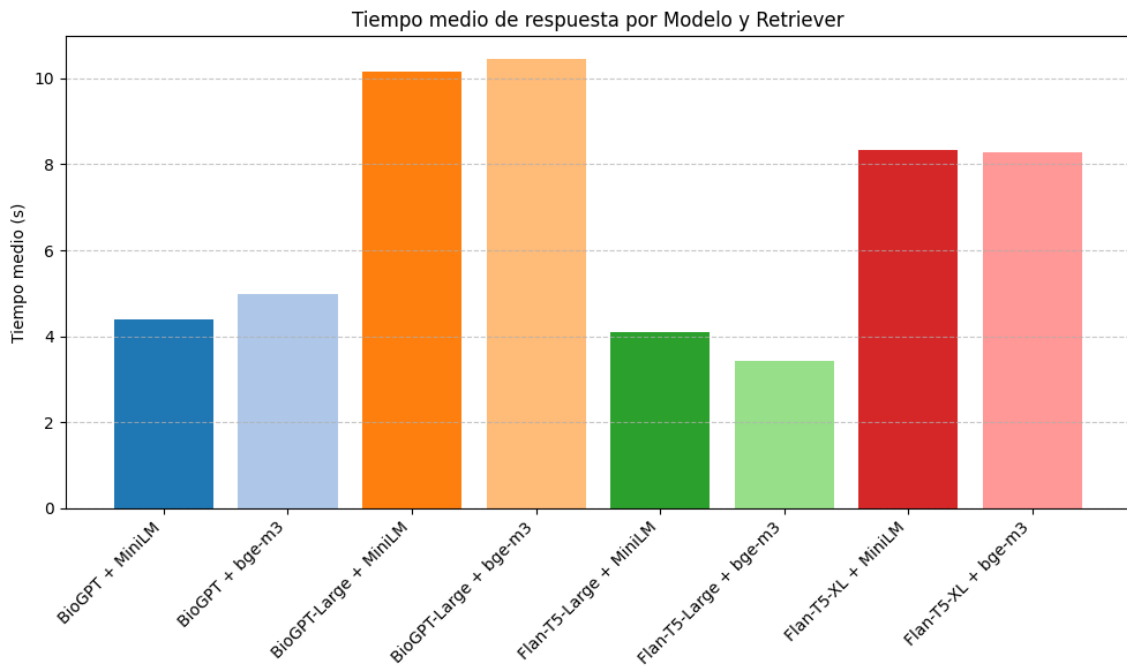


Figura 6.28: Tiempo medio (s) por modelo y retriever empleado

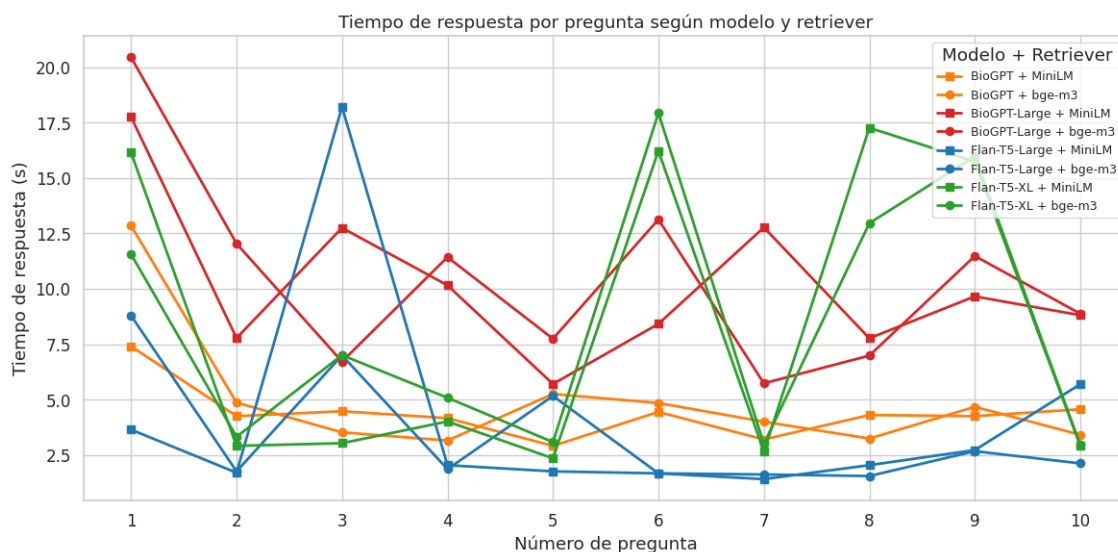


Figura 6.29: Tiempo (s) por pregunta, modelo y retriever empleado

De los gráficos anteriores se pueden extraer las siguientes conclusiones:

- **Modelos de embeddings (retrievers):** el tiempo medio de respuesta no parece estar significativamente influenciado por el modelo de *embeddings* utilizado. Las diferencias entre *bge-m3* y *MiniLM* no superan el segundo en la mayoría de los casos. Además, su rendimiento relativo varía dependiendo del *LLM* con el que se integren, siendo *bge-m3* más rápido en unos casos y *MiniLM* en otros. Por tanto, se concluye que la recuperación de documentos no representa un cuello de botella en la latencia del sistema.
- **Modelos generativos (LLM):** como se observa en la figura 6.28, los modelos con mayor número de parámetros, tanto de propósito general (**Flan-T5-XL**) como especializados en el ámbito biomédico (**BioGPT-Large**), presentan tiempos de respuesta más elevados, lo cual es coherente debido a sus mayores requisitos computacionales. Además, según el gráfico de líneas (figura 6.29), se aprecia que los modelos **BioGPT** muestran una mayor estabilidad en sus respuestas, mientras que los modelos **Flan-T5** presentan una variabilidad mayor, con picos pronunciados entre preguntas consecutivas.

6.4.4. Escalabilidad

La escalabilidad es un objetivo el cual no se puede evaluar directamente mediante métricas numéricas. Sin embargo, la propuesta empleada para el chatbot *RAG* sí demuestra que es un sistema escalable debido a la fácil implementación y gestión de los artículos y expansión del corpus documental gracias al uso de la herramienta **ChromaDB** en conjunto con **Langchain**.

ChromaDB permite que la base de datos pueda ser reutilizada de forma sencilla y, mediante la función propia **from_documents()**, permite la adición de documentos sin la necesidad de reconstruir el sistema completo.

```
1 #Listado de documentos a insertar
2 documents=[...] #listado de documentos a insertar
3 vectordb = Chroma.from_documents(
4     documents=documents,
5     #Modelo de embeddings a emplear
6     embedding=embedding_model,
7     #Ruta de almacenamiento local
8     persist_directory="/content/drive/MyDrive/AN_NO_STRUCT/TFM/rare_DB_bg3"
```

Listing 6: Código necesario para añadir documentos a ChromaDB

En el caso de que se publiquen nuevos artículos en la revista newsRARE, solo necesitará seguir los siguientes pasos:

1. **Scraping y preprocesamiento**
2. **Vectorización mediante el modelo de embedding seleccionado (p.ej. bge-m3)**
3. **Inserción en la ruta local donde se encuentra persistida la base de datos**

Además, como el sistema emplea un modelo *LLM* preentrenado desacoplado del almacenamiento vectorial, no será necesaria la reimplementación del modelo o la regeneración de los *embeddings* anteriores. Gracias a esto, se ve significativamente reducido el coste computacional y garantiza que las respuestas relativas a contenido anterior mantengan su validez.

6.4.5. Conclusión de la evaluación del sistema RAG

A raíz de la evaluación realizada, se puede concluir que el modelo **BioGPT**, junto al retriever (modelo de *embeddings*) **bge-m3**, ofrece el mejor rendimiento en términos de **veracidad** desde una perspectiva tanto cuantitativa, obteniendo las mejores métricas, como cualitativa, al generar respuestas fieles al contexto aportado sin alucinaciones. Respecto a la **robustez del sistema**, la herramienta ha demostrado una alta capacidad para identificar preguntas que se encuentren fuera del dominio planteado, gracias al uso adecuado de *prompt engineering*, especialmente a través de la plantilla nº1 (Listing 3), la cual garantiza una interacción segura con el usuario.

En términos de **velocidad**, se ha comprobado que, a pesar de la mayor carga computacional asociada a algunos de los modelos empleados, los tiempos de respuesta se mantienen dentro de márgenes razonables, los cuales no van a afectar a la experiencia del usuario de forma negativa. Finalmente, en lo relativo a la **escalabilidad**, el diseño modular basado en **ChromaDB** y **LangChain** permite que el sistema sea capaz de incorporar nuevos documentos de forma eficiente y sin la necesidad de rehacer pasos previos generando nuevo código.

Por lo tanto, se puede afirmar que el sistema generado, lejos de ser la herramienta de chatbot óptima, cumple con los objetivos iniciales planteados al inicio del proyecto, convirtiéndose en una aplicación sólida y funcional que puede aportar gran valor y funcionalidad a la página web newsRARE.

7. Conclusiones, limitaciones y trabajos futuros

El desarrollo de este proyecto ha permitido el diseño, implementación y evaluación de un sistema basado en la recopilación y procesamiento de textos del portal **newsRARE**, el análisis temático, la generación automática de resúmenes y la creación de un chatbot, basado en una arquitectura *RAG*, orientado a proporcionar respuestas verificadas en el dominio de las enfermedades raras.

7.1. Limitaciones halladas durante el proyecto

A pesar de los resultados satisfactorios obtenidos en los diferentes bloques del proyecto, es necesario reconocer las diferentes limitaciones halladas en cada una de las partes del proyecto.

- **Adaptabilidad del sistema:** el sistema se ha diseñado específicamente para el tratamiento de los artículos del corpus **newsRARE**. El uso de este sistema empleando otras fuentes de información puede generar resultados muy dispares a los hallados en este proyecto.
- **Traducción automática:** debido a la falta de modelos biomédicos entrenados directamente en español en los ámbitos de *summarization* como de *text-generation*, se ha optado por un proceso de traducción y retraducción automática. Aunque este proceso ha mejorado la comprensión de los modelos generativos, puede introducir imprecisiones léxicas o pérdidas de matices importantes.
- **Evaluación cuantitativa limitada:** aunque se han aplicado métricas automáticas relevantes como *ROUGE-L*, *BERTScore-F1* y *Similitud Coseno*, la evaluación cuantitativa se ha llevado a cabo sin disponer de un conjunto de datos de referencia con etiquetas validadas, lo cual refleja que, al tratarse de un enfoque no supervisado, ha sido necesario aplicar procedimientos de evaluación alternativos que no garantizan una valoración óptima de los temas y textos generados.
- **Evaluación cualitativa limitada:** la evaluación cualitativa no ha sido realizada por personal experto en enfermedades raras, lo que restringe la fiabilidad clínica de las observaciones obtenidas y limita la capacidad para generalizar los resultados a un contexto profesional o sanitario.
- **Limitaciones computacionales:** el despliegue de modelos más potentes (como **BioGPT-Large**) queda limitado por el entorno gratuito utilizado (**Google Colab**), lo que impide extraer las capacidades de estos modelos al máximo.

7.2. Líneas de trabajo futuro

A partir de estas limitaciones, se proponen diversas líneas de trabajo que podrían reforzar, escalar o diversificar el sistema desarrollado:

- **Evaluación con la colaboración de expertos:** llevar a cabo una evaluación exhaustiva mediante técnicas de *crowdsourcing* a expertos del dominio médico, que permita contrastar la validez de las respuestas a nivel clínico y semántico.
- **Optimización del pipeline multilingüe:** explorar nuevos modelos biomédicos entrenados directamente en español, como por ejemplo [RoBERTa-bio-es](#), o desarrollar un fine-tuning de modelos existentes generalistas para mejorar el rendimiento sin necesidad de procesos de traducción.
- **Incorporación de feedback del usuario:** introducir mecanismos de retroalimentación por parte de los usuarios del sistema que permitan refinar de forma iterativa el rendimiento de la aplicación con el uso de técnicas propias del *reinforcement learning with human feedback* [29].
- **Despliegue en entorno real:** convertir el prototipo en una herramienta integrada dentro del sitio web de *newsRARE*, optimizando la experiencia de usuario, incorporando interfaces adaptativas y garantizando la protección de datos **GDPR**[10]. Para ello, las tecnologías para almacenar los diferentes módulos del sistema podrían ser las siguientes:
 - **ChromaDB y LLM:** almacenar tanto los modelos en proveedor cloud como GCP (Google Cloud Platform) o **AWS** (Amazon Web Services).
 - **Gradio App:** la aplicación con la interfaz se puede almacenar en un contenedor **Docker** y sería desplegada en el servidor cloud deseado.

El proyecto realizado supone una base sólida enfocada para la creación de herramientas en el dominio del procesamiento del lenguaje natural. A pesar de las limitaciones mostradas, el sistema demuestra un gran potencial y las técnicas empleadas de *RAG*, *topic modeling* y automatización de resúmenes han facilitado la creación de un sistema no solo permite estructurar de mejor forma el contenido de la revista *newsRARE*, sino también impulsar la labor divulgativa de la revista, fomentando el acceso a estos contenidos mediante los resúmenes automatizados y a involucrar a los usuarios a que consulten sus dudas y curiosidades a través del *chatbot*.

Por lo tanto, se concluye que los objetivos propuestos al principio de este proyecto han sido alcanzados de manera satisfactoria y que la arquitectura planteada presenta múltiples vías de desarrollo que podrán aportar mejoras en la veracidad, escalado y fiabilidad del sistema.

Referencias

- [1] K. R. Chowdhary. “Natural Language Processing”. En: *Fundamentals of Artificial Intelligence*. New Delhi: Springer India, 2020, págs. 603-649. ISBN: 978-81-322-3972-7. DOI: [10.1007/978-81-322-3972-7_19](https://doi.org/10.1007/978-81-322-3972-7_19). URL: https://doi.org/10.1007/978-81-322-3972-7_19.
- [2] Ashish Vaswani et al. “Attention is All You Need”. En: *Advances in neural information processing systems*. Vol. 30. Curran Associates, Inc. 2017.
- [3] Konstantinos I Roumeliotis y Nikolaos D Tselikas. “Chatgpt and open-ai models: A preliminary review”. En: *Future Internet* 15.6 (2023), pág. 192.
- [4] Mr D Murahari Reddy et al. “Dall-e: Creating images from text”. En: *UGC Care Group I Journal* 8.14 (2021), págs. 71-75.
- [5] Y Yogi Tegar Nugroho y P Paulus Metta Dwi Manggala. “The Use of AI in Creating Music Compositions: A Case Study on Suno Application”. En: *7th Celt International Conference (CIC 2024)*. Atlantis Press. 2024, págs. 177-189.
- [6] Tomas Mikolov et al. “Efficient Estimation of Word Representations in Vector Space”. En: *arXiv preprint arXiv:1301.3781* (2013).
- [7] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. En: *arXiv preprint arXiv:1810.04805* (2018).
- [8] Yanshan Wang et al. “Clinical information extraction applications: a literature review”. En: *Journal of Biomedical Informatics* 77 (2018), págs. 34-49.
- [9] Patrick Lewis et al. “Retrieval-augmented generation for knowledge-intensive NLP tasks”. En: *Advances in Neural Information Processing Systems* 33 (2020), págs. 9459-9474.
- [10] EU GDPR. *General data protection regulation (gdpr)*. 2018.
- [11] Denis Rothman. *Transformers for Natural Language Processing: Build, train, and fine-tune deep neural network architectures for NLP with Python, Hugging Face, and OpenAI's GPT-3, ChatGPT, and GPT-4*. Packt Publishing Ltd, 2022.
- [12] Oguzhan Topsakal y Tahir Cetin Akinci. “Creating large language model applications utilizing langchain: A primer on developing LLM apps fast”. En: *International Conference on Applied Engineering and Natural Sciences*. Vol. 1. 1. 2023, págs. 1050-1056.
- [13] *ChromaDB*. URL: <https://www.trychroma.com/>.
- [14] *Sentence Transformers*. URL: <https://sbert.net/>.
- [15] Fabian Pedregosa et al. “Scikit-learn: Machine learning in Python”. En: *The Journal of Machine Learning Research* 12 (2011), págs. 2825-2830.
- [16] J. D. Hunter. “Matplotlib: A 2D Graphics Environment”. En: *Computing in Science amp; Engineering* (2007).
- [17] Alex Johnson et al. *Plotly Graphing Libraries*. URL: <https://plotly.com/graphing-libraries/>.

REFERENCIAS

- [18] Abubakar Abid et al. “Gradio: Hassle-free sharing and testing of ml models in the wild”. En: *arXiv preprint arXiv:1906.02569* (2019).
- [19] Anish Chapagain. *Hands-On Web Scraping with Python: Perform advanced scraping operations using various Python libraries and tools such as Selenium, Regex, and others*. Packt Publishing Ltd, 2019.
- [20] Travis E. Oliphant. *Guide to NumPy*. USA: Trelgol Publishing, 2006.
- [21] Wes McKinney. *Python for Data Analysis*. California, USA: O’Reilly, 2013.
- [22] Steven Bird. “NLTK: the natural language toolkit”. En: *Proceedings of the COLING/ACL 2006 interactive presentation sessions*. 2006, págs. 69-72.
- [23] Yuli Vasiliev. *Natural language processing with Python and spaCy: A practical introduction*. No Starch Press, 2020.
- [24] Prafulla Bafna, Dhanya Pramod y Anagha Vaidya. “Document clustering: TF-IDF approach”. En: *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*. IEEE. 2016, págs. 61-66.
- [25] Julian Eggert y Edgar Korner. “Sparse coding and NMF”. En: *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541)*. Vol. 4. IEEE. 2004, págs. 2529-2533.
- [26] Laurens van der Maaten y Geoffrey Hinton. “Visualizing data using t-SNE”. En: *Journal of machine learning research* 9.Nov (2008), págs. 2579-2605.
- [27] Michael Röder, Andreas Both y Alexander Hinneburg. “Exploring the space of topic coherence measures”. En: *Proceedings of the eighth ACM international conference on Web search and data mining*. 2015, págs. 399-408.
- [28] Manik Bhandari et al. “Re-evaluating evaluation in text summarization”. En: *arXiv preprint arXiv:2010.07100* (2020).
- [29] Yuntao Bai et al. “Training a helpful and harmless assistant with reinforcement learning from human feedback”. En: *arXiv preprint arXiv:2204.05862* (2022).

ANEXO A: Detalles de implementación y código

A.1 Implementación Scraper

```
1 !pip install bs4
2 !pip install requests
3
4 import pandas as pd
5 from unicode import unicode
6 import json
7 import requests
8
9 from selenium import webdriver
10 from selenium.webdriver.chrome.service import Service
11 from selenium.webdriver.common.by import By
12 from selenium.webdriver.chrome.options import Options
13 from bs4 import BeautifulSoup
14 import time
15
16 """# Scraping de los artículos de NewsRare por secciones (Selenium +BS4)
17
18 En primer lugar tenemos que poder acceder a cada sección
19 """
20
21 import requests
22 from bs4 import BeautifulSoup
23
24 # Descargar la página
25 url = "https://newsrare.es/articulos"
26 response = requests.get(url)
27 # Parsear el HTML
28 soup = BeautifulSoup(response.text, "html.parser")
29 # Encontrar el bloque de secciones
30 section_container = soup.find("div", class_="td_block_categories_tags")
31 # Extraer los enlaces de las secciones
32 sections = section_container.find_all("a")
33
34 sections_urls=[]
35 nombre_seccion=[]
36 num_articulos=[]
37
38 for content in sections:
39
40     section_url=content["href"]
41     name=content.find("span",class_="td-ct-item-name").get_text()
42     num=content.find("span",class_="td-ct-item-no").get_text()
43
44     sections_urls.append(section_url)
45     nombre_seccion.append(name)
46     num_articulos.append(num)
47
48 df_secciones=pd.DataFrame({
```

Anexo A

```
49     "Seccion":nombre_seccion,
50     "N_Articulos":num_articulos,
51     "URL":sections_urls
52 })
53
54 df_secciones.head()
55
56 """Acceder a los artículos de cada sección
57
58 # Obtención de artículos por seccion ( Es necesario usar Selenium para hacer
59   scroll en la página y que cargen todos)
60 """
61 def obtener_articulos_seccion(url_seccion,filepath): # ej: filepath ./articles
62     # Selenium configuration
63     chrome_options = Options()
64     chrome_options.add_argument("--headless") # Run in headless mode (no UI)
65     chrome_options.add_argument("--disable-gpu")
66     chrome_options.add_argument("--no-sandbox")
67
68     # Path to WebDriver (Modify if needed)
69     webdriver_path = "./chromedriver.exe"
70
71     # Start WebDriver
72     service = Service(webdriver_path)
73     driver = webdriver.Chrome(service=service, options=chrome_options)
74
75     # URL de la página de "Actualidad"
76     #url_actualidad = "https://newsrare.es/seccion/actualidad/"
77     # Cargar la página
78     driver.get(url_seccion)
79     category=url_seccion.split("/")[-2]
80     filename="noticias_"+category+".json"
81
82     # Espera para asegurar que la página cargue completamente
83     time.sleep(3)
84
85     # Realizar scroll para cargar más artículos
86     scroll_pause_time = 15 # Tiempo de espera entre desplazamientos
87     last_height = driver.execute_script("return document.body.scrollHeight")
88
89     while True:
90         # Hacer scroll hasta el fondo de la página
91         driver.execute_script("window.scrollTo(0, document.body.scrollHeight);"
92                                )
93         time.sleep(scroll_pause_time)
94
95         # Verificar si el scroll alcanzó el final de la página
96         new_height = driver.execute_script("return document.body.scrollHeight")
97         if new_height == last_height:
98             break
99         last_height = new_height
100
101     # Una vez que la página ha cargado completamente, obtener el contenido HTML
102     soup = BeautifulSoup(driver.page_source, "html.parser")
```

Anexo A

```
103 # Buscar los artículos en la categoría 'Actualidad'
104 articles = soup.find_all("div", class_="td-module-meta-info")
105
106 # Filtrar los artículos cuya categoría sea 'Actualidad'
107 counter=0
108 title_set=set()
109 noticias_list=[]
110 for article in articles:
111     category_element = article.find("a", class_="td-post-category")
112     category_element_clean = unicode(category_element.get_text(strip=True)
113     ).lower().replace(" ", "-")
114     if category_element and category in category_element_clean:
115         a_element = article.find("a")
116         h3_element = article.find("h3")
117         url_article=h3_element.find("a")["href"]
118         title=h3_element.get_text(strip=True)
119         if title not in title_set: #chequeo de duplicados
120             counter+=1
121             title_set.add(title) # Add title to the set
122             #print(f"{counter}. {a_element.get_text(strip=True)} | {title}
123             | URL: {url_article}")
124             noticias_list.append({
125                 "title": title,
126                 "url": url_article
127             })
128
129 # Cerrar el navegador de Selenium después de terminar
130 driver.quit()
131
132 #Creamos json con todas las noticias
133 with open(filepath+"/"+filename, "w") as file:
134     json.dump(noticias_list, file)
135 print("JSON Creado Correctamente: ",filename)
136
137 df_secciones["URL"]
138 for url in df_secciones["URL"].values:
139     print(url)
140     filename=url.split("/")[-2]
141     print (filename)
142
143 for url in df_secciones["URL"]:
144     obtener_articulos_seccion(url, "./articles")
145
146 """"## Obtención del texto para los artículos para todas las secciones""""
147
148 import os
149 import json
150 import requests
151 from bs4 import BeautifulSoup
152
153 articles_folder_path = "./articles"
154
155 def get_article_text(url):
156     """"Obtener el texto de un artículo de una URL""""
157     try:
158         # Realizar la solicitud HTTP para obtener el HTML de la página
159         response = requests.get(url)
```

```
157
158     if response.status_code == 200:
159         # Parsear el HTML con BeautifulSoup
160         soup = BeautifulSoup(response.text, 'html.parser')
161
162         # Acceder a la info del artículo
163         article_body = soup.find_all(class_="elementor-widget-container")
164         article_text = ""
165         for section in article_body:
166             article_text += section.text + "\n"
167
168         if article_text == "": # El cuerpo del artículo puede estar en
169             # otra clase
170             article_body = soup.find_all(class_="td-post-content")
171             for section in article_body:
172                 article_text += section.text + "\n"
173
174         return article_text.strip() # Eliminar espacios en blanco extra
175
176     else:
177         print(f"Error al acceder a la URL: {url}")
178         return None
179
180 except Exception as e:
181     print(f"Error al obtener el artículo de {url}: {e}")
182     return None
183
184 def update_json_files(folder_path):
185     """Actualizar los archivos JSON con el texto del artículo"""
186     for filename in os.listdir(folder_path):
187         if filename.endswith('.json'):
188             file_path = os.path.join(folder_path, filename)
189
190             with open(file_path, 'r', encoding='utf-8') as f:
191                 articles = json.load(f)
192
193             # Actualizar cada entrada con el texto del artículo
194             for article in articles:
195                 url = article.get("url")
196                 if url:
197                     print(f"Procesando artículo: {article['title']}")
198                     article_text = get_article_text(url)
199                     if article_text: # Evitar asignar valores vacíos o None
200                         article["content"] = article_text
201
202             # Guardar el archivo JSON actualizado (indentación corregida)
203             with open(file_path, 'w', encoding='utf-8') as f:
204                 json.dump(articles, f, ensure_ascii=False, indent=4)
205                 print(f"Archivo {filename} actualizado.")
206
207 # Llamar a la función para actualizar los archivos
208 update_json_files(articles_folder_path)
```

Listing 7: Código completo Scraper

A.2 Implementación Topic Modeling

A.2.1 – Preprocesamiento del texto y análisis exploratorio

```
1 import re
2 import pandas as pd
3 import spacy
4 import nltk
5 from nltk.corpus import stopwords
6 from wordcloud import WordCloud
7 from collections import Counter
8 import matplotlib.pyplot as plt
9
10 # Descargar recursos necesarios
11 nltk.download('stopwords')
12 !python -m spacy download es_core_news_sm
13 nlp = spacy.load("es_core_news_sm")
14
15 # Lista de stopwords en español
16 stop_words = set(stopwords.words('spanish'))
17
18 def clean_text(text):
19     """Limpieza básica del texto."""
20     text = re.sub(r"\n", " ", text)
21     text = re.sub(r"\s+", " ", text)
22     text = re.sub(r"[^a-zA-ZáéíóúÁÉÍÓÚñÑü\ſ]", "", text)
23     return text.strip().lower()
24
25 def lemmatize_text(text):
26     """Lematización del texto usando spaCy."""
27     doc = nlp(text)
28     return " ".join([token.lemma_ for token in doc if token.lemma_ not in
29                     stop_words and not token.is_punct])
30
31 def preprocess_column(df, col_name):
32     """Aplica limpieza y lematización a una columna de texto."""
33     df[col_name] = df[col_name].astype(str).apply(clean_text)
34     df[f"processed_{col_name}"] = df[col_name].apply(lemmatize_text)
35     return df
36
37 def generate_wordcloud(text, title=""):
38     """Genera e imprime una nube de palabras a partir de texto."""
39     wordcloud = WordCloud(width=800, height=400, background_color="white").
40         generate(text)
41     plt.figure(figsize=(10, 5))
42     plt.imshow(wordcloud, interpolation="bilinear")
43     plt.axis("off")
44     plt.title(title)
45     plt.show()
46
47 def get_term_frequencies(text_series, top_n=20):
48     """Devuelve las palabras más frecuentes de una serie de texto."""
49     all_words = " ".join(text_series).split()
```

```
48 return Counter(all_words).most_common(top_n)
```

Listing 8: Preprocesamiento del texto y análisis exploratorio

A.2.2 – Vectorización del corpus y NMF

```
1
2 from sklearn.feature_extraction.text import TfidfVectorizer
3 from sklearn.decomposition import NMF
4
5 def apply_nmf(data, column, size_ngram=1, num_topics=10, n_top_words=6,
6 stop_words=None):
7     """
8     Aplica NMF a una columna de texto del DataFrame.
9
10    Parámetros:
11    - data: DataFrame con los textos.
12    - column: nombre de la columna a utilizar.
13    - size_ngram : tamaño del n-grama (1->palabra, 2-> bigrama, 3->trigrama)
14    - num_topics: número de tópicos a extraer.
15    - n_top_words: número de palabras clave por tópico.
16    - stop_words: lista de stopwords personalizada.
17
18    Devuelve:
19    - DataFrame actualizado con columnas de tópicos asignados.
20    - Diccionario con palabras clave por tópico.
21    - Modelo NMF entrenado.
22    - Matriz TF-IDF.
23    """
24    vectorizer = TfidfVectorizer(ngram_range=(size_ngram, size_ngram), max_df=
25    =0.3, min_df=2, stop_words=stop_words)
26    tfidf_matrix = vectorizer.fit_transform(data[column])
27
28    nmf_model = NMF(n_components=num_topics, random_state=42, max_iter=500)
29    nmf_model.fit(tfidf_matrix)
30
31    feature_names = vectorizer.get_feature_names_out()
32    topic_words = {}
33    for topic_idx, topic in enumerate(nmf_model.components_):
34        top_words_indices = topic.argsort()[::-n_top_words - 1:-1]
35        topic_words[topic_idx] = [feature_names[i] for i in top_words_indices]
36
37    assigned_topics = nmf_model.transform(tfidf_matrix).argmax(axis=1)
38    data[f'assigned_{column}_topic'] = assigned_topics
39    data[f'topic_{column}_words'] = data[f'assigned_{column}_topic'].apply(
40        lambda topic_idx: ", ".join(topic_words[topic_idx])
41    )
42    return data, topic_words, nmf_model, tfidf_matrix
```

Listing 9: TF-IDF y uso de NMF sobre columna del dataframe deseada

A.2.3 – Visualización de tópicos con t-SNE y Plotly

```
1
2 from sklearn.manifold import TSNE
3 import plotly.graph_objects as go
4 import pandas as pd
5
6 def apply_tsne(tfidf_matrix, perplexity=30, n_iter=1000, random_state=42):
7     """
8     Aplica t-SNE a una matriz TF-IDF o documento-tema (como la W de NMF).
9
10    Parámetros:
11    - tfidf_matrix: matriz de características (e.g., salida de NMF.transform())
12    - perplexity: parámetro de t-SNE para control de densidad local.
13    - n_iter: número de iteraciones.
14
15    Devuelve:
16    - Coordenadas 2D reducidas como un array NumPy.
17    """
18    tsne = TSNE(n_components=2, perplexity=perplexity, n_iter=n_iter,
19               random_state=random_state)
20    tsne_coords = tsne.fit_transform(tfidf_matrix)
21    return tsne_coords
22
23 def plot_tsne_with_topics(tsne_coords, topics, hover_text=None, title="
24 Distribución de artículos por temas"):
25     """
26     Visualiza los resultados de t-SNE en 2D con colores por tópico asignado.
27
28     Parámetros:
29     - tsne_coords: matriz de coordenadas 2D reducidas.
30     - topics: lista de tópicos asignados a cada punto.
31     - hover_text: lista opcional de textos a mostrar al pasar el ratón.
32     """
33     df_vis = pd.DataFrame({
34         "x": tsne_coords[:, 0],
35         "y": tsne_coords[:, 1],
36         "topic": topics,
37         "text": hover_text if hover_text is not None else [f"Doc {i}" for i in
38                   range(len(topics))]
39     })
40
41     fig = go.Figure()
42
43     for topic in sorted(df_vis["topic"].unique()):
44         topic_data = df_vis[df_vis["topic"] == topic]
45         fig.add_trace(go.Scatter(
46             x=topic_data["x"],
47             y=topic_data["y"],
48             mode='markers',
49             name=f"Tópico {topic}",
50             text=topic_data["text"],
51             hoverinfo='text',
```

```

49         marker=dict(size=8, opacity=0.7)
50     ))
51
52     fig.update_layout(
53         title=title,
54         xaxis_title="t-SNE Dim 1",
55         yaxis_title="t-SNE Dim 2",
56         template="plotly_white",
57         showlegend=True
58     )

```

Listing 10: Visualizado de topics con t-SNE y Plotly

A.3 Implementación Resumen Automatizado

A.3.1 – Implementación de herramienta (enfoque traducción intermedia)

```

1  import gradio as gr
2  import textwrap
3  import requests
4  from bs4 import BeautifulSoup
5  from transformers import pipeline
6
7  # Modelos de traducción y resumen
8  traductor = pipeline("translation_es_to_en", model="Helsinki-NLP/opus-mt-es-en")
9  summarizer = pipeline("summarization", model="Falconsai/medical_summarization")
10 de_traductor = pipeline("translation_en_to_es", model="Helsinki-NLP/opus-mt-en-es")
11
12 #División de los artículos en chunks de tamaño fijo
13 def split_text(text, max_length=512):
14     return textwrap.wrap(text, max_length)
15
16 #traducción español-inglés
17 def translate_text(text):
18     chunks = split_text(text)
19     return " ".join([traductor(c)[0]["translation_text"] for c in chunks])
20
21 #traducción inglés-español
22 def de_translate_text(text):
23     chunks = split_text(text)
24     return " ".join([de_traductor(c)[0]["translation_text"] for c in chunks])
25
26 # Aplicar el modelo de resumen a los n primeros chunks del artículo
27 def summarize_text(text):
28     chunks = split_text(text)
29     summarized = [summarizer(c, max_length=100, min_length=40)[0]["summary_text"] for c in chunks[:1]]
30     return " ".join(summarized)

```

```

31
32 # Aplicar Scraping del artículo a partir de la url introducida por el usuario
33 def get_article_text(url):
34     try:
35         response = requests.get(url)
36         if response.status_code == 200:
37             soup = BeautifulSoup(response.text, 'html.parser')
38             article_text = ""
39             title_tag = soup.find(class_="entry-title")
40             title = title_tag.get_text(strip=True) if title_tag else (soup.
41                 title.string.strip() if soup.title else "Sin título")
42             article_body = soup.find_all(class_="elementor-widget-container")
43                 or soup.find_all(class_="td-post-content")
44             for section in article_body:
45                 article_text += section.get_text(separator=" ").strip() + "\n"
46             return article_text.strip() or "[ERROR] Artículo vacío.", title
47         else:
48             return f"[ERROR] Código de respuesta HTTP: {response.status_code}",
49                 "Error"
50     except Exception as e:
51         return f"[ERROR] {e}", "Error"
52
53 # Generar la publicación en función de la red social indicada por el usuario
54 def generar_publicacion(texto, titulo, url, red):
55     resumen = summarize_text(translate_text(texto))
56     resumen_es = de_translate_text(resumen)
57
58     if red == "Twitter (X)":
59         limite = 280
60     elif red == "LinkedIn":
61         limite = 3000
62     elif red == "Instagram":
63         limite = 2200
64     else:
65         limite = 1000
66
67     header = f"{titulo}\n "
68     footer = f"\n{url}"
69     otros = len(header) + len(footer)
70     espacio_resumen = limite - otros
71
72     frases = resumen_es.strip().split(" ")
73     resumen_final = ""
74     total_len = 0
75     for frase in frases:
76         if not frase.endswith("."):
77             frase += "."
78         if total_len + len(frase) <= espacio_resumen:
79             resumen_final += frase + " "
80             total_len += len(frase) + 1
81         else:
82             break
83     if not resumen_final:
84         resumen_final = resumen_es[:espacio_resumen].rsplit(" ", 1)[0]
85
86     post = header + resumen_final.strip() + footer

```

```

84     return resumen_es, post
85
86 # Crear pipeline que agrupe todas las funciones necesarias a aplicar para
87 # emplear en la interfaz de Gradio
88 def pipeline_resumen(url, red):
89     texto, titulo = get_article_text(url)
90     if texto.startswith("[ERROR]"):
91         return texto, "", ""
92     resumen, post = generar_publicacion(texto, titulo, url, red)
93     return texto, resumen, post
94 # Implementación en Gradio previamente indicada en la sección 5.2.3 (Listing 2)

```

Listing 11: Implementación de herramienta (enfoque traducción intermedia)

Cabe destacar que el **enfoque directo**, que emplea un modelo que trata textos en castellano, utiliza un código muy similar al anterior donde la única diferencia se da en que no se lleva a cabo el proceso de traducción al inglés y retraducción al castellano tras resumir el texto del artículo obtenido.

A.4 Implementación Chatbot RAG

A.4.1 – Creación de base de datos vectorial (embeddings)

```

1 import pandas as pd
2 from langchain.text_splitter import RecursiveCharacterTextSplitter
3 from langchain.docstore.document import Document
4 from langchain.embeddings import HuggingFaceEmbeddings
5 from langchain.vectorstores import Chroma
6
7 # 1. Cargar el dataset de artículos
8 df = pd.read_csv("../all_articles_with_source.csv")
9 df['text'] = df['text'].fillna('') # limpieza de nulos
10
11 # 2. Dividir el texto en chunks con solapamiento.
12 text_splitter = RecursiveCharacterTextSplitter(
13     chunk_size=500,
14     chunk_overlap=100,
15     separators=["\n\n", "\n", ".", " ", "", ""]
16 )
17
18 documents = []
19 for _, row in df.iterrows():
20     base_metadata = {
21         "title": row['title'],
22         "url": row['article_url']
23     }
24     split_texts = text_splitter.split_text(row['text'])
25

```

```

26     for chunk_idx, text in enumerate(split_texts):
27         metadata = base_metadata.copy()
28         metadata["chunk_index"] = f"{row['article_url']}#chunk_{chunk_idx}"
29         documents.append(Document(page_content=text, metadata=metadata))
30
31 # 3. Definir el modelo de embeddings a utilizar
32 embedding_model = HuggingFaceEmbeddings(model_name="sentence-transformers/all-
    MiniLM-L6-v2")
33
34 # 4. Almacenar los documentos en ChromaDB empleando el modelo de embeddings
    seleccionado
35 vectordb = Chroma.from_documents(
36     documents=documents,
37     embedding=embedding_model,
38     persist_directory="./rare_DB_MiniLM"
39 )
40
41 print(f"Ingsted {len(documents)} chunks with embeddings into ChromaDB.")

```

Listing 12: Proceso de creación de base de datos vectorial en ChromaDB

A.4.2 –Integración completa de los elementos del Chatbot RAG

```

1
2 import os
3 import gradio as gr
4 from transformers import AutoTokenizer, AutoModelForSeq2SeqLM, pipeline
5 from langchain_chroma import Chroma
6 from langchain_huggingface import HuggingFaceEmbeddings
7 from langchain.chains import RetrievalQA
8 from langchain.prompts import PromptTemplate
9 from langchain_community.llms import HuggingFacePipeline
10
11 # 1. Cargar el modelo de embeddings
12 embedding_model = HuggingFaceEmbeddings(model_name="BAAI/bge-m3")
13
14 # 2. Cargar la base de datos vectorial persistida (ChromaDB)
15 vectorstore = Chroma(
16     persist_directory="/content/drive/MyDrive/AN_NO_STRUCT/TFM/rare_DB_bg3",
17     embedding_function=embedding_model
18 )
19
20 # 3. Crear el recuperador semántico a partir de la base vectorial
21 retriever = vectorstore.as_retriever(search_kwargs={"k": 3})
22
23 # 4. Definir la plantilla de prompt que restringe la generación al contexto
    recuperado
24 custom_prompt = PromptTemplate(
25     input_variables=["context", "question"],
26     template=(
27         "Responde únicamente con la información contenida en el contexto
            proporcionado.\n"

```

```

28     "Si no puedes responder con el contexto, simplemente indica: 'No se
        encontró suficiente información en la base de datos para responder
        a esta pregunta.'\n\n"
29     "Contexto:\n{context}\n\n"
30     "Pregunta: {question}\n\n"
31     "Respuesta:"
32 )
33 )
34
35 # 5. Cargar el modelo generativo (en este caso, FLAN-T5-Large)
36 tokenizer = AutoTokenizer.from_pretrained("google/flan-t5-large")
37 model = AutoModelForSeq2SeqLM.from_pretrained("google/flan-t5-large")
38 pipe = pipeline("text2text-generation", model=model, tokenizer=tokenizer,
        max_new_tokens=250)
39 llm = HuggingFacePipeline(pipeline=pipe)
40
41 # 6. Crear la cadena RAG que integra recuperación + generación
42 qa_chain = RetrievalQA.from_chain_type(
43     llm=llm,
44     retriever=retriever,
45     chain_type="stuff",
46     return_source_documents=True,
47     chain_type_kwargs={"prompt": custom_prompt}
48 )
49
50 # 7. Definir la función de interacción del chatbot con recuperación de fuentes
51 def responder_chat(message, history):
52     try:
53         resultado = qa_chain.invoke({"query": message})
54         respuesta = resultado["result"]
55         docs = resultado.get("source_documents", [])
56         fuentes = "\n\n".join(
57             [f"{doc.metadata.get('title', 'Sin título')}: \n{doc.page_content
                    [:400]}..." for doc in docs]
58         )
59         return f"{respuesta}\n\nFragmentos recuperados: \n{fuentes}"
60     except Exception as e:
61         return f"Error en la generación de respuesta: {str(e)}"
62
63 # 8. Crear la interfaz conversacional con Gradio (ChatInterface)
64 with gr.Blocks() as demo:
65     with gr.Row():
66         with gr.Column():
67             chatbot = gr.ChatInterface(
68                 fn=responder_chat,
69                 type="messages",
70                 title="RAREbot - Chatbot RAG sobre Enfermedades Raras",
71                 flagging_mode="manual",
72                 flagging_options=["Like", "Spam", "Inappropriate", "Other"],
73                 save_history=True,
74             )
75
76 # 9. Lanzar la aplicación web
77 demo.launch(share=True, debug=True)

```

Listing 13: Integración completa de los elementos del Chatbot RAG empleando LangChain