Universidad Zaragoza
1542

Trabajo Fin de Grado

Buscador y clasificador de noticias sobre la agricultura

Agriculture news search and classifier

Autor

David Zandundo Fuster

Director

Jorge Laguna Argüello

Ponente

Javier Lacasta Miguel

Escuela de Ingeniería y Arquitectura

Ingeniería del Software

2022/2023

BUSCADOR Y CLASIFICADOR DE NOTICIAS SOBRE LA AGRICULTURA

RESUMEN

En los últimos años, se ha observado un incremento exponencial en la cantidad de información relacionada con el sector agrícola disponible en la red. Este crecimiento se debe al rápido desarrollo de los medios digitales y las redes sociales. Sin embargo, dicha información se encuentra dispersa en una amplia variedad de fuentes y formatos, lo que dificulta su procesamiento y análisis por parte de los interesados en el campo agrícola. Esta situación puede representar un desafío significativo, ya que contar con información actualizada y precisa es fundamental para la toma de decisiones acertadas en áreas como la gestión de recursos, el control de plagas y enfermedades, y la mejora de la productividad y la calidad de los cultivos.

En este contexto, el presente TFG plantea el diseño y la implementación de un buscador y clasificador de noticias especializado en el ámbito agrícola. Este buscador y clasificador tiene como objetivo recopilar y estructurar de manera eficiente la información disponible en la red sobre agricultura. Para lograrlo, se utilizarán herramientas y tecnologías avanzadas en los campos de la ciencia de datos y el procesamiento de lenguaje natural. Esta solución permitirá la extracción de datos relevantes provenientes de diversas fuentes, su adaptación a un formato común y su almacenamiento en una base de datos centralizada. De esta manera, los usuarios podrán acceder y analizar posteriormente esta información de manera más eficiente y precisa.

Además, este buscador y clasificador de noticias sobre agricultura facilitará la tarea de mantenerse actualizado en un campo tan dinámico. Los usuarios podrán acceder de manera rápida y sencilla a información relevante, como avances tecnológicos en el sector agrícola, investigaciones científicas, políticas y regulaciones vigentes, tendencias de mercado, consejos prácticos y casos de éxito en la industria.

ABSTRACT

In recent years, there has been an exponential increase in the amount of agricultural-related information available on the internet. This growth is due to the rapid development of digital media and social networks. However, this information is scattered across a wide variety of sources and formats, making it difficult for stakeholders in the agricultural field to process and analyze. This situation can pose a significant challenge, as having up-to-date and accurate information is crucial for making informed decisions in areas such as resource management, pest and disease control, and improving crop productivity and quality.

In this context, this project proposes the design and implementation of a specialized agricultural news search engine and classifier. The goal of this search engine and classifier is to efficiently gather and structure information available on the internet related to agriculture. To achieve this, advanced tools and technologies in the fields of data science and natural language processing will be utilized. This solution will enable the extraction of relevant data from diverse sources, its adaptation to a common format, and its storage in a centralized database. In this way, users will be able to access and subsequently analyze this information more efficiently and accurately.

Furthermore, this agricultural news search engine and classifier will facilitate the task of staying updated in such a dynamic field. Users will be able to quickly and easily access relevant information, such as technological advancements in the agricultural sector, scientific research, current policies and regulations, market trends, practical tips, and success stories in the industry.

DECLARACIÓN DE AUTORÍA

DECLARACIÓN DE RESPONSABILIDAD Y AUTORÍA DEL TFG:

D.: David Zandundo Fuster, con DNI 73163787M, estudiante del Grado de Ingeniería informática de la Universidad de Zaragoza y tutelado por D.: Jorge Laguna Argüello

DECLARO QUE:

El Trabajo Fin de Grado denominado “Buscador y clasificador de noticias sobre la agricultura” ha sido desarrollado respetando la propiedad intelectual (citando las fuentes bibliográficas utilizadas en la redacción de dicho trabajo), así como cualquier otro derecho, por ejemplo el de imagen que pudiese estar sujeto a protección del copyright.

En virtud de esta declaración afirmo que este trabajo es inédito y de mi autoría, por lo que me responsabilizo del contenido, veracidad y alcance del Trabajo Fin de Grado, y asumo las consecuencias administrativas y jurídicas que se deriven en caso de incumplimiento de esta declaración.

Y para que así conste, firmo la presente declaración en Zaragoza, a 6 de Junio de 2023.

Fdo.: 

Este documento formará parte de la memoria del TFG correspondiente.

ÍNDICE

Buscador y clasificador de noticias sobre la agricultura 2
Resumen 2
Abstract 3
Declaración de autoría 4
Índice de ilustraciones 7
Índice de tablas 8
Agradecimientos 10
1 Introducción 11
1.1 Contexto del trabajo 11
1.2 Contexto tecnológico y herramientas utilizadas 11
1.3 Alcance, objetivos y limitaciones 13
1.4 Esquema general de la memoria del proyecto 14
2 Diseño general del sistema 15
2.1 Requisitos generales del sistema 15
2.2 Propuesta arquitectural conceptual 15
2.3 Propuesta arquitectural tecnológica 17
3 Implementación del sistema 18
3.1 Capa de extracción, transformación y almacenamiento de datos 18
3.2 Gestión del flujo de trabajo 19
3.3 Indexación 20
3.3.1 Modelos de embeddings generalista 21
3.3.2 Modelos de embeddings especializados 24
3.4 Revisor de noticias 27
3.4.1 Control de accesibilidad 27
3.4.2 Control de la actualización del contenido 27
3.5 API 28
3.6 Interfaz web 29
3.6.1 Diseño de la interfaz 30
4 Conclusiones y trabajo futuro 31

4.1 Conocimientos adquiridos 31
4.2 Trabajo futuro 31
4.3 Conclusiones 32
Lecturas Recomendadas 34
Anexo I. Un poco más del prototipo 35
Tecnologías más en profundidad 35
Diseño del sistema más detallado 38
Estructura y organización del sistema a nivel de código 42
Diseño y técnicas sobre la indexación 43
Motivos para la elección de la API Deepl 44
Anexo II. Resultados obtenidos 46
Resultados para elegir el score 52
Anexo III. Problemas sin solución 54
Etiquetas desbalancean los resultados 54
Descripciones cuando el contenido había cambiado 54

ÍNDICE DE ILUSTRACIONES

Ilustración 1: Stack de tecnologías usado en el TFG. 13
Ilustración 2: Diagrama de arquitectura conceptual del sistema. 16
Ilustración 3: Diagrama de arquitectura a nivel tecnológico del sistema. 17
Ilustración 4: Diagrama de modelo relacional de la entidad noticia. 18
Ilustración 5: DAG que siguen los procesos de ETL. 19
Ilustración 6: Diagrama de la memoria ocupada con los diferentes modelos generalistas. 22
Ilustración 7: Gráfica con los tiempos de ejecución para cada modelo generalista. 23
Ilustración 8: Diagrama de la memoria ocupada con los diferentes modelos especializados junto con el modelo generalista que tiene menor consumo de memoria. 25
Ilustración 9: Gráfica con los tiempos de ejecución de todos los modelos. 25
Ilustración 10: Gráfica con los datos de precisión, real y k-precisión para cada modelo. 26
Ilustración 11: Interfaz web final. 30
Ilustración 12: Diagrama de secuencia de los módulos que componen el ETL. 39
Ilustración 13: Diagrama de secuencia de los módulos que componen el indexador. 40
Ilustración 14: Diagrama de secuencia de los módulos que componen el revisor. 41

ÍNDICE DE TABLAS

Listado 1: Estructura de carpetas y ficheros. 42

Tabla 1: Consultas realizadas. 46

Tabla 2: Resultados de la consulta 1. 47

Tabla 3: Resultados de la consulta 2. 48

Tabla 4: Resultados de la consulta 3. 48

Tabla 5: Resultados de la consulta 4. 49

Tabla 6: Resultados de la consulta 5. 49

Tabla 7: Resultados de la consulta 6. 49

Tabla 8: Resultados de la consulta 7. 50

Tabla 9: Resultados de la consulta 8. 50

Tabla 10: Resultados de la consulta 9. 51

Tabla 11: Resultados generales calculados a partir de las consultas. 51

Tabla 12: Resultados de la consulta 'Cereales' para obtener el parámetro score. 52

Tabla 13: Resultados de la consulta 'Pistachos' para obtener el parámetro score. 53

Tabla 14: Resultados de la consulta 'Maíz' para obtener el parámetro score. 53

Esta página ha sido intencionalmente dejada en blanco

AGRADECIMIENTOS

En primer lugar, deseo expresar mi más sincero agradecimiento a mis profesores, Javier Lacasta Miguel y Francisco Javier Zarazaga, así como a mis estimados compañeros, Jorge Laguna Argüello y Víctor Jarreta Espligares, por su dedicación y orientación durante el proceso de tutoría y desarrollo de este proyecto. Su experiencia y conocimiento me han guiado de manera excepcional, brindándome valiosas referencias que han resultado fundamentales para el éxito de este trabajo.

En segundo lugar, me gustaría expresar mi profunda gratitud hacia mis compañeros universitarios, quienes han sido mis compañeros de viaje en este largo, pero fructífero recorrido por el mundo de la informática. Su colaboración ha sido fundamental para alcanzar una comprensión más profunda de los temas abordados y, como resultado, he experimentado un notable crecimiento en mis habilidades y conocimientos.

Asimismo, quisiera aprovechar esta oportunidad para agradecer a la Universidad de Zaragoza por proporcionarme los recursos y conocimientos necesarios a lo largo de estos años para la realización exitosa de este Trabajo Final de Grado. La calidad de la educación recibida en esta institución ha sido fundamental para mi formación académica y el logro de este importante hito en mi carrera.

Por último, pero no por ello menos importante, deseo expresar mi más profundo agradecimiento a mi familia y amigos, quienes han sido un pilar fundamental en todo momento. Su apoyo moral inquebrantable ha sido de vital importancia para la realización de este proyecto.

1 INTRODUCCIÓN

1.1 Contexto del trabajo

La agricultura tradicional ha formado parte de la actividad humana desde tiempos antiguos, en los que las tareas en el campo eran realizadas manualmente. Con el paso del tiempo, se empezó a contar con la ayuda de animales para llevar a cabo las tareas más arduas, y posteriormente, las máquinas fueron incorporadas a la actividad agrícola con el fin de aligerar el esfuerzo humano en las labores de cultivo. Actualmente, la tendencia natural es delegar cada vez más dichas tareas a máquinas y sistemas automatizados, siempre bajo la supervisión y dirección de los seres humanos, y con la asistencia de sistemas inteligentes que permiten tomar decisiones óptimas, mejorando así la eficiencia y reduciendo los costes asociados a la producción agrícola.

En los últimos años, se ha observado un incremento exponencial en la cantidad de información útil para dichos sistemas automáticos disponible en la red, lo cual ha sido posible gracias al crecimiento vertiginoso de los medios digitales y las redes sociales. No obstante, dicha información se encuentra dispersa en una amplia variedad de fuentes y formatos, lo que dificulta en gran medida su procesamiento y análisis por parte de los interesados en la materia. Este hecho puede resultar particularmente problemático en un sector como la agricultura, en el que la información actualizada y precisa puede ser crucial para la toma de decisiones acertadas en áreas como la gestión de recursos, el control de plagas y enfermedades, y la mejora de la productividad y la calidad de los cultivos.

En este contexto, el diseño y la implementación de un buscador y clasificador de noticias sobre la agricultura se presenta como una solución eficiente para recopilar y estructurar la información disponible en la red sobre este sector. Un buscador y clasificador permite la extracción de datos de distintas fuentes, su transformación para adaptarlos a un formato común y su carga en una base de datos centralizada, que puede ser consultada y analizada posteriormente.

Este trabajo pretende ser una contribución al ámbito de la agricultura y la tecnología, al facilitar el acceso a información actualizada y estructurada sobre este sector, lo que puede mejorar la toma de decisiones de los actores involucrados en la cadena productiva agrícola.

1.2 Contexto tecnológico y herramientas utilizadas

En la actualidad existen múltiples tecnologías y herramientas disponibles para el desarrollo de un buscador y clasificador de noticias. Estas tecnologías y herramientas han sido creadas para permitir el procesamiento y la transformación de grandes volúmenes de datos de manera eficiente.

La elección de las herramientas y tecnologías adecuadas para el desarrollo de un buscador y clasificador es una tarea importante y crítica que debe basarse en su eficacia y madurez en el ámbito de la ciencia de datos y el procesamiento de lenguaje natural. Para lograr esto, se

deben considerar diferentes factores, como la escalabilidad, la facilidad de uso, la flexibilidad y el soporte para la integración con otras herramientas.

Requerimos de herramientas que permitan la extracción de información de manera directa desde la web, así como la transformación de dichos datos en una estructura adecuada para su almacenamiento en una base de datos. Además, necesitamos implementar un sistema de indexación que facilite la realización de búsquedas sobre estos datos. Asimismo, será necesario establecer un mecanismo periódico para ejecutar las tareas de extracción, transformación y carga (ETL). Por último, se precisa desarrollar una API que sirva como punto de conexión al cual pueda conectarse una interfaz web, facilitando así la realización de solicitudes de manera más intuitiva.

Es en este contexto donde se hace necesaria la utilización de herramientas tecnológicas especializadas en la extracción de datos web, se ha elegido la herramienta Beautiful Soup. Se trata de una biblioteca de Python, que permite la extracción de datos de archivos HTML y XML de forma sencilla y rápida. Esta herramienta es capaz de navegar por la estructura de una página web y extraer la información requerida, ya sea texto, imágenes o cualquier otro elemento presente en el código fuente.

En el marco de nuestro proyecto, se había identificado la necesidad de contar con una herramienta que permitiera orquestar y automatizar workflows de manera eficiente y efectiva. Ante esta situación, se llevó a cabo un análisis detallado de las opciones disponibles en el mercado, considerando diversos criterios como su capacidad de integración con otras herramientas, su escalabilidad, su facilidad de uso, entre otros aspectos. Después de un proceso de evaluación riguroso, se llegó a la conclusión de que Apache Airflow era la mejor opción para nuestras necesidades específicas.

En lo que respecta a las herramientas de procesamiento de datos y bases de datos, se ha llevado a cabo una elección cuidadosa y detallada, con el objetivo de asegurar la utilización de herramientas que se ajusten perfectamente a las necesidades específicas del proyecto. En este sentido, se ha decidido utilizar Pandas y MongoDB, dos herramientas de renombre mundial que han demostrado ser muy populares y efectivas en la manipulación y gestión de grandes cantidades de datos.

En primer lugar, Pandas es una biblioteca de Python que se especializa en el análisis de datos y que proporciona una amplia variedad de herramientas para la manipulación y transformación de datos en un formato tabular. Con Pandas, se pueden realizar fácilmente operaciones de limpieza de datos, como la eliminación de duplicados, la normalización de datos y la conversión de tipos de datos, entre otras.

Por otro lado, MongoDB es una base de datos NoSQL[1] que se orienta a documentos, lo que significa que permite el almacenamiento de grandes cantidades de datos de manera flexible y escalable.

[1] NoSQL es un enfoque de almacenamiento y recuperación de datos que no utiliza tablas relacionales y permite una alta escalabilidad y flexibilidad en la estructura de los datos.

Esta herramienta se caracteriza por ser altamente escalable, lo que permite una rápida expansión en la capacidad de almacenamiento y procesamiento de datos. Además, MongoDB es una herramienta altamente flexible, lo que permite la gestión de datos en diferentes formatos, lo que resulta en una gran ventaja en entornos donde los datos pueden provenir de diferentes fuentes y tener diferentes formatos.

En el contexto de nuestro proyecto, se presentaba la necesidad de contar con una herramienta capaz de indexar y almacenar documentos de manera eficiente, con el objetivo de facilitar su posterior búsqueda y recuperación. Para cumplir con este requerimiento, se llegó a la conclusión de que ElasticSearch era la solución más adecuada para nuestras necesidades específicas. ElasticSearch es una herramienta de búsqueda y análisis distribuida, diseñada para proporcionar una búsqueda de alta velocidad y una recuperación de datos eficiente.

Ilustración 1: Stack de tecnologías usado en el TFG.

Por último, para la implementación del sistema, se utilizan herramientas de programación como Python y sus librerías, así como herramientas de integración continua y despliegue continuo, como Git. Para obtener una descripción más detallada de las tecnologías mencionadas, se recomienda consultar el anexo I.

1.3 Alcance, objetivos y limitaciones

El objetivo de este trabajo es el de construir una primera versión de un sistema de extracción de noticias sobre la agricultura de diversas fuentes en línea, como periódicos y revistas especializadas. La extracción de los datos debe estar automatizada. Su clasificación, indexación y búsqueda se basará en técnicas de procesamiento de lenguaje natural.

Al tratarse de una primera versión, se ha limitado el alcance de este TFG a un conjunto inicial de cuatro portales de noticias sobre agricultura: Europapress[2] (sección de Agricultura), Heraldo de Aragón[3] (sección del Campo), Ministerio de Agricultura y Pesca[4] (página oficial del ministerio), y Revista Agricultura[5] (todas las secciones).

Se espera que en un futuro el sistema pueda ser escalado para incluir muchos más portales de noticias. Finalmente, el alcance del buscador también será muy acotado dado que su único papel es el de actuar como elemento demostrador de funcionalidad, pero meramente accesorio.

1.4 Esquema general de la memoria del proyecto

La presente memoria del proyecto se encuentra estructurada de la siguiente manera:
En la sección 2, se proporciona una detallada descripción del análisis y diseño del buscador y clasificador de noticias. Se presentan los requisitos, así como la arquitectura general y tecnológica empleada.

En la sección 3, se aborda el desarrollo del sistema, poniendo especial énfasis en las decisiones de implementación adoptadas y los desafíos que surgieron durante el proceso.

La sección 4 se dedica a exponer las lecciones aprendidas y las conclusiones derivadas del trabajo realizado. Asimismo, se plantean proyectos futuros y funcionalidades próximas a ser incorporadas al sistema.

En el anexo I, se encuentra disponible una explicación completa y detallada acerca de las tecnologías empleadas en el presente proyecto. Además, se brinda información adicional que abarca el diseño de cada componente que conforma el sistema, ofreciendo un entendimiento más profundo de su funcionamiento y estructura.

El anexo II está dedicado a una explicación sumamente detallada de los resultados obtenidos una vez que el sistema estuvo en funcionamiento. Se realiza una comparativa de los resultados alcanzados con otras posibles decisiones que podrían haber sido tomadas.

Por último, en el anexo III se abordan los desafíos y problemáticas encontrados que, hasta el momento, no han sido resueltos. Se exponen las soluciones más apropiadas para cada uno de estos problemas, ofreciendo así una visión integral y propositiva hacia posibles resoluciones en el futuro.

[2] Enlace al portal de noticias de Europapress. https://www.europapress.es/temas/agricultura/

[3] Enlace al portal de noticias del Heraldo de Aragón.
https://www.heraldo.es/tags/temas/heraldo_del_campo.html

[4] Enlace al portal de noticias del Ministerio de Agricultura y Pesca.
https://www.mapa.gob.es/es/prensa/ultimas-noticias/

[5] Enlace al portal de noticias de Revista Agricultura. https://www.revistaagricultura.com

2 DISEÑO GENERAL DEL SISTEMA

2.1 Requisitos generales del sistema

El proyecto tiene tres objetivos primordiales que se deben satisfacer. En primer lugar, se requiere un servicio de extracción de nuevas noticias, que seguirá un patrón fijo establecido y que podrá ser configurable. El sistema accede a los diferentes portales de noticias y extrae las últimas noticias posteriores a una fecha fijada en el sistema. Una vez extraídas, las noticias serán guardadas en la base de datos, para luego ser analizadas y clasificadas.

En segundo lugar, se requiere un sistema de gestión que permita mostrar listados completos de todas las noticias almacenadas en la base de datos, permitiendo su ordenación, filtrado y borrado. Además, el sistema permitirá cambiar la fecha de ejecución de los procesos de extracción, así como la fecha inicial de búsqueda de noticias.

En tercer lugar, se implementará un buscador que permite la búsqueda de noticias en la base de datos a través de rangos de fechas y palabras clave. Los resultados obtenidos por el buscador podrán ser ordenados y filtrados por diferentes campos. La implementación de esta herramienta dotará al sistema de una capacidad de búsqueda avanzada y personalizada, lo que permitirá una gestión eficiente y una rápida recuperación de la información relevante para los usuarios.

Por último, se llevará a cabo la implementación de un demonio[6] encargado de la limpieza de las URL perdidas. Este proceso consistirá en una revisión periódica de todas las URL indexadas, con el fin de asegurar su accesibilidad. En el caso de que alguna de estas URL no se encuentre disponible, se realizarán N intentos para comprobar su acceso, y en caso de que la URL siga sin ser accesible, se procederá a su eliminación de manera automática. Esta funcionalidad garantizará la eficacia y la calidad de la información ofrecida por el sistema, y permitirá una gestión óptima del contenido indexado.

2.2 Propuesta arquitectural conceptual

Con las anteriores funcionalidades, se plantea la arquitectura del sistema que queremos construir y podemos observar cómo actúan los componentes del sistema. En la Ilustración 2 se puede visualizar un diagrama arquitectural a un nivel conceptual, centrado en la arquitectura clásica de un ETL. Tenemos, por un lado, una capa externa al sistema que se compone de todos los portales web de noticias al que el sistema va a acceder para extraer la información. En medio se encuentra nuestro sistema, que ofrece al exterior una API para poder realizar consultas y acceder al sistema. Por último, se ofrece una interfaz web que se conecta a la API anteriormente mencionada para así facilitar al usuario hacer consultas al sistema.

[6] Un demonio es un programa o proceso que se ejecuta en segundo plano de forma continua, y se encarga de realizar tareas específicas o proporcionar servicios a otros programas.

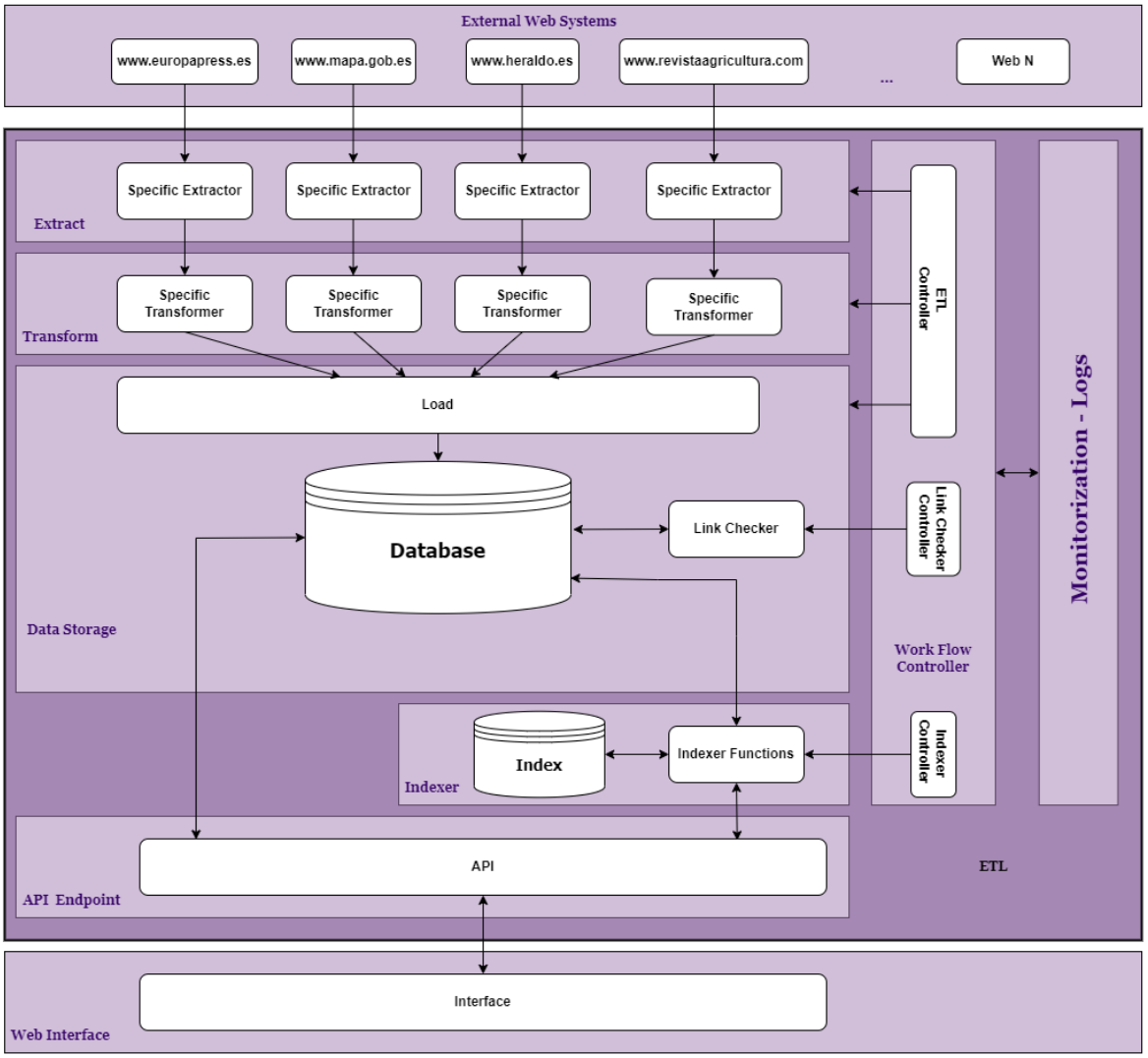

Ilustración 2: Diagrama de arquitectura conceptual del sistema.

Observando la ilustración 2, podemos apreciar que el sistema se encuentra dividido en diversas capas, cada una de las cuales desempeña una función específica en el proceso de gestión de la información. En la capa"Extract", se lleva a cabo la extracción de datos desde las distintas páginas web, y para ello se dispone de un adaptador particular para cada sitio web, ya que la información se encuentra ubicada en diferentes secciones según el portal.

Posteriormente, en la capa "Transform", se produce la transformación de los datos, para lo cual se cuenta con un módulo específico para cada sitio web, que se encarga de realizar las transformaciones necesarias para que los datos extraídos tengan una estructura uniforme en todos los portales.

A continuación, tenemos la capa "Data Storage", compuesta por el módulo de carga de datos y el módulo de revisión de accesibilidad de enlaces. Tras ser transformados, los datos son almacenados en la base de datos y se verifica periódicamente que los enlaces asociados a las noticias continúen siendo accesibles.

La capa "Indexer" se encarga de la indexación de las noticias que se han procesado previamente según el modelo de embeddings escogido. Por separado, se encuentra la capa "API Endpoint" previamente mencionada, que se encargará de conectar la interfaz web con las funcionalidades del sistema. Por último, se encuentra la capa de gestión de flujos de trabajo "Work Flow Controller", la cual se ejecutará de manera constante y se encargará de supervisar el correcto funcionamiento de todas las capas anteriores. Además, en esta capa se almacenarán los datos de monitorización y registros para su posterior revisión.

2.3 Propuesta arquitectural tecnológica

Una vez que se ha analizado el sistema a nivel conceptual, se procederá a examinar la arquitectura en su nivel tecnológico. En la sección 1.2 se presentaron las tecnologías que han sido empleadas en el sistema, y ahora se mostrarán en el diagrama de la ilustración 3 para una mejor comprensión y visualización del mismo.

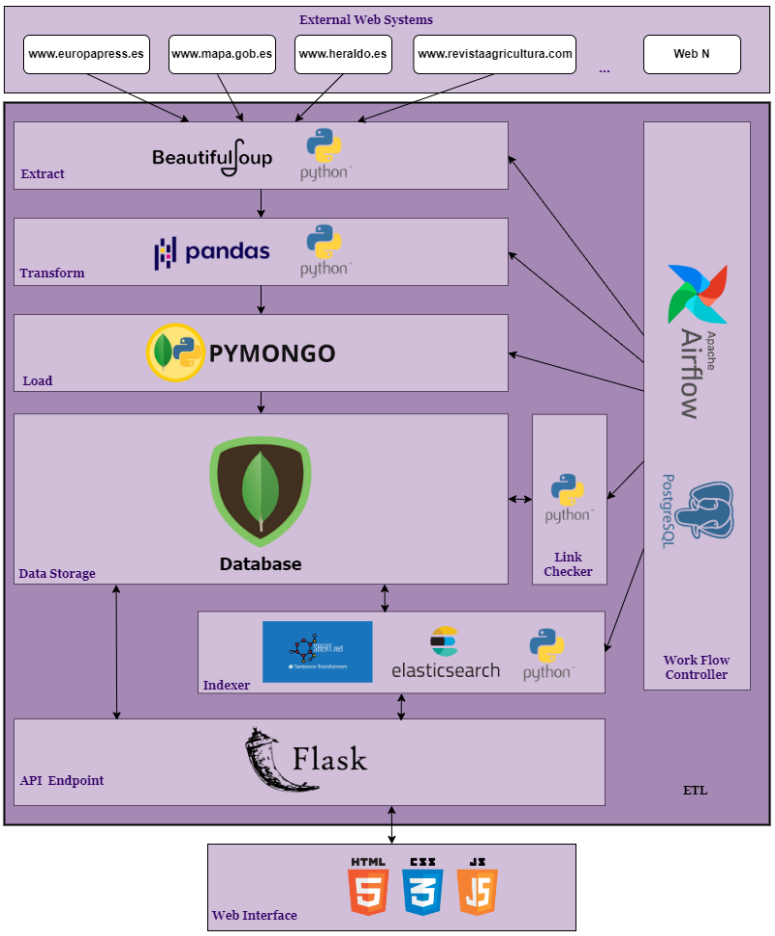

Ilustración 3: Diagrama de arquitectura a nivel tecnológico del sistema.

En lo que respecta al despliegue, se realiza de manera fácil y sencilla mediante el encapsulamiento de todos los componentes que se visualizan en el diagrama de la Ilustración 3 en contenedores de Docker. Este proceso se lleva a cabo a través de un archivo "docker-compose.yml", que permite la creación y gestión de los diferentes contenedores de manera automatizada y eficiente. En el anexo 1 se proporciona una explicación detallada de los componentes de esta arquitectura y de cómo interactúan entre sí.

3 IMPLEMENTACIÓN DEL SISTEMA

En el siguiente apartado, se llevará a cabo una subdivisión en las diversas capas previamente mencionadas, donde se abordarán con mayor detalle los desafíos que surgieron en cada una de ellas, así como las respectivas soluciones implementadas.

3.1 Capa de extracción, transformación y almacenamiento de datos

Inicialmente, se procedió a identificar los atributos que conforman una noticia y que serían útiles para su posterior diferenciación, indexación y búsqueda. Se acordó que una noticia se compondrá principalmente de un título, una descripción, un texto, una fecha de publicación, el nombre del portal al que pertenece y una lista de etiquetas asociadas. Esto se puede apreciar con mayor claridad en la ilustración 4.

Con el fin de simplificar el almacenamiento de las noticias, se tomó la decisión de asignar a cada una un identificador único en forma de cadena de caracteres. MongoDB proporciona esta funcionalidad de manera predeterminada al agregar un documento[7] a una colección.

Desde un principio se observó que no todas las noticias presentaban los mismos atributos de manera constante. Por ejemplo, en algunos portales se encontraban etiquetas asociadas a las noticias, mientras que en otros portales estas etiquetas no existían. Sin embargo, al utilizar una base de datos NoSQL como MongoDB, fue posible mitigar este problema gracias a su naturaleza flexible.

Noticia		
PK	ID	
Unique Key	Tiíulo	*type:varchar*
	Texto	*type:varchar*
	Fecha	*type:date*
	Fuente	*type:varchar*
	Descripción	*type:varchar*
	Enlace	*type:varchar*
Nullable	Etiquetas	*type:list(varchar)*

Ilustración 4: Diagrama del modelo relacional de la entidad noticia.

[7] En MongoDB, un documento es la unidad básica de almacenamiento de datos y representa una entidad en forma de un conjunto de pares clave-valor. Una colección es un grupo lógico de documentos que comparten una estructura similar y se almacenan juntos en la base de datos.

Posteriormente se identificó que existían noticias provenientes del mismo portal con contenido idéntico, pero con enlaces diferentes, lo cual resultaba en la creación de noticias separadas a pesar de compartir la misma información (título, descripción, texto). Para abordar esta situación, se estableció una clave única compuesta por el título, texto, fecha y fuente de la noticia. De esta manera, nos aseguramos de evitar la inclusión de noticias con contenido duplicado.

3.2 Gestión del flujo de trabajo

Una vez que se ha completado el desarrollo de las funciones de ETL adaptadas a cada portal, es necesario programar su ejecución en intervalos de tiempo específicos, teniendo en cuenta la frecuencia de publicación de noticias de cada portal. Para ello se ha elegido el controlador de flujo de trabajo Apache Airflow.

Apache Airflow emplea grafos acíclicos dirigidos[8] (DAG) para la orquestación del flujo de trabajo. Los DAG pueden programarse para ejecutarse en horarios predeterminados, como por ejemplo cada hora o cada día. Las tareas y sus dependencias se definen en Python, y posteriormente Airflow se encarga de la planificación y ejecución de las tareas.

La solución más adecuada es la creación de un DAG individualizado para cada portal, programándolos para su ejecución en horarios distintos. Sin embargo, esta solución plantea varios inconvenientes. En primer lugar, el sistema presenta una escasa capacidad de escalabilidad, ya que la adición de nuevos portales requiere de la creación de un nuevo DAG desde cero para cada uno, lo cual afecta significativamente al rendimiento y dificulta su desarrollo e integración. En segundo lugar, los parámetros de entrada de un DAG en Apache Airflow no son dinámicos, lo que impide modificar la fecha de recogida de las noticias en cada ejecución agendada.

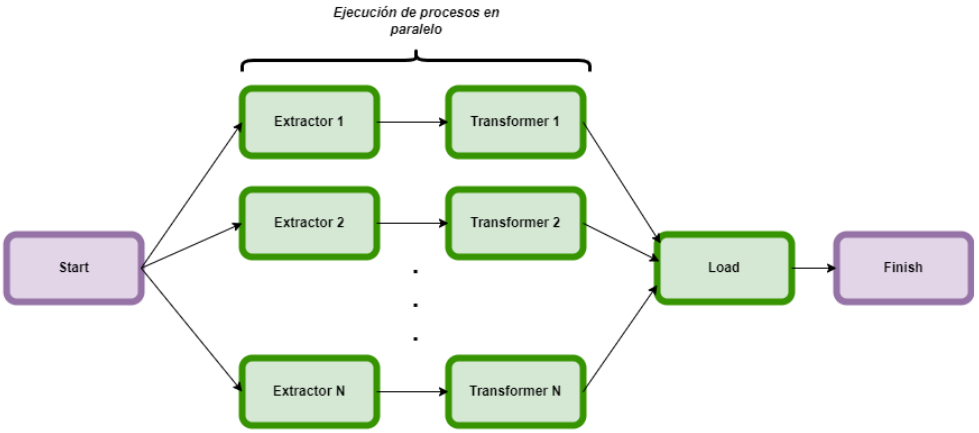

Ilustración 5: DAG que siguen los procesos de ETL.

[8] DAG son estructuras de datos que consisten en un conjunto de nodos interconectados por flechas que indican la dirección de las relaciones entre ellos. La característica principal de un DAG es que no contiene ciclos, lo que significa que no hay caminos que forman bucles cerrados.

Con el fin de abordar esta situación, se decidió implementar un enfoque consistente en el desarrollo de un único DAG que se activaría y ejecutaría diariamente, como se muestra en la ilustración 5. De este modo, el mencionado DAG al activarse se divide en un número determinado de procesos, equivalente al total de portales presentes en nuestro sistema. Cada uno de estos procesos tendría la responsabilidad de verificar si corresponde ejecutar las tareas de ETL para su portal respectivo. En caso afirmativo, una vez completadas las tareas de ETL, se actualizará la fecha de la última ejecución exitosa con la fecha actual. Por eso, también es necesario almacenar en la base de datos la fecha de la última ejecución exitosa de las funciones de ETL correspondientes a cada portal, junto con la frecuencia de ejecución asociada.

Lo ideal también sería indexar todas las noticias cada vez que se llevan a cabo las tareas de almacenamiento en la base de datos. Sin embargo, como se explicará detalladamente en la sección 3.3, resulta impracticable mantener la mayor parte de la memoria ocupada constantemente por un modelo de embeddings. Para abordar esta limitación, se ha implementado un enfoque similar al mencionado anteriormente para la indexación de las noticias. En este nuevo enfoque, cada vez que se introduce una noticia en la base de datos, su identificador se guarda en una lista. Semanalmente, nuestro indexador recupera dicha lista y procede a indexar todos los documentos identificados por los identificadores contenidos en la lista. De esta manera, evitamos tener ocupada casi en su totalidad la memoria con el modelo en todo momento.

3.3 Indexación

Cuando se terminó de implementar las funciones ETL y se comprobó que el controlador del flujo de trabajo funcionaba correctamente, se procedió a pensar cómo se iban a indexar las noticias para poder implementar el buscador.

En primer lugar, se planteó la necesidad de determinar el enfoque a seguir para la indexación y búsqueda de documentos. Se consideró inicialmente la opción más sencilla, que consiste en la indexación y búsqueda en texto plano[9]. Sin embargo, se aspiraba a que nuestro sistema pudiera ofrecer noticias estrechamente relacionadas con la consulta en caso de no encontrar resultados exactos. Por ejemplo, si un usuario desea buscar noticias relacionadas con "almendra" y no hay ninguna noticia que contenga esa palabra en la base de datos, se buscaba que el sistema ofreciera noticias relacionadas con avellanas, cacahuetes, castañas, y otros términos afines. Se evaluaron diversas técnicas, como se detalla en el anexo I, sin embargo, se tomó la decisión final de utilizar embeddings[10] debido a una serie de consideraciones. Los embeddings ofrecen la capacidad de representar de manera numérica las relaciones y similitudes entre palabras y documentos.

[9] La búsqueda en texto plano es un proceso de encontrar ocurrencias de una cadena de texto dentro de un texto más largo. Se realiza comparando la cadena de búsqueda con el texto en busca de coincidencias exactas.

[10] Los embeddings son representaciones vectoriales de palabras o frases que capturan el significado y la semántica en un espacio n-dimensional.

Esta técnica permite capturar y aprovechar el contexto semántico y la estructura subyacente de los datos, lo que conduce a resultados de búsqueda más precisos y relevantes. Además, los embeddings proporcionan una mayor flexibilidad y capacidad para realizar búsquedas semánticas, incluso en casos en los que las consultas no coinciden exactamente con los términos presentes en los documentos indexados.

Entre las tecnologías consideradas que pudieran soportar este enfoque se incluyeron soluciones de indexación de alto rendimiento, como motores de búsqueda basados en Apache Lucene, Elasticsearch y Solr. Después de un análisis de las diferentes tecnologías de indexación disponibles, se decidió usar Elasticsearch debido a su capacidad de manejar grandes volúmenes de datos, escalabilidad, flexibilidad y robustez. La elección de Elasticsearch se basó en su rendimiento eficiente, soporte de consultas avanzadas, estabilidad comprobada y una comunidad activa.

3.3.1 Modelos de embeddings generalistas

Una vez definidas la tecnología y la técnica a utilizar, se procedió a la búsqueda de un modelo de embeddings que pudiera proporcionar resultados satisfactorios. Un modelo de embeddings es esencialmente un diccionario de palabras en el que cada palabra está asociada a un vector de N componentes de números reales. Estos vectores capturan las relaciones semánticas y contextuales entre las palabras, de manera que palabras con un contexto y significado similar tendrán vectores con valores cercanos entre sí.

Se encontraron varios que fueron entrenados de manera diferente:

- Modelo 1: Modelo de 985.667 vectores con 300 componentes cada uno, entrenado con el algoritmo de FastText[11] con datos de la wikipedia en español. Con un tamaño total de 2.2 GB.
- Modelo 2: Modelo de 1.313.423 vectores con 300 componentes, entrenado con el algoritmo de GloVe[12] con los datos SBWC[13]. Con un tamaño total de 3.4 GB
- Modelo 3: Modelo de 1.313.423 vectores con 300 componentes, entrenado con el algoritmo de FastText con los datos SUC[14]. Con un tamaño total de 2.9 GB

Como se puede apreciar, la adopción de este enfoque implicaba la carga en memoria de una gran cantidad de datos. Con el objetivo de reducir el tamaño de los modelos, se consideraron diferentes estrategias. En este sentido, surgió la idea de utilizar el algoritmo PCA[15] para disminuir la dimensionalidad de los vectores y, en consecuencia, reducir de manera significativa el tamaño de los modelos.

[11] FastText es un algoritmo de aprendizaje automático utilizado para representar palabras como vectores densos y realizar clasificación de texto. https://fasttext.cc

[12] GloVe es un enfoque de aprendizaje automático que utiliza matrices de coocurrencia de palabras para generar representaciones vectoriales densas de palabras.

[13] SBWC (Spanish Billion Word Corpus). https://crscardellino.ar/SBWCE/

[14] SUC (Spanish Unannotated Corpora). https://github.com/josecannete/spanish-corpora

[15] PCA (Principal Component Analysis) es una técnica de reducción de dimensionalidad que busca transformar un conjunto de variables correlacionadas en un conjunto de nuevas variables no correlacionadas llamadas componentes principales, que explican la mayor varianza de los datos. https://es.wikipedia.org/wiki/Análisis_de_componentes_principales

Esta técnica permitiría mantener una representación más compacta de los datos, preservando al mismo tiempo la mayor parte de su variabilidad y características distintivas. Es por esto que se añadió un nuevo modelo a nuestra colección de pruebas, partiendo del modelo de mayor tamaño, en vista de la suposición de que proporcionaría los mejores resultados.

- Modelo 4: Modelo de 1.313.423 vectores con 100 componentes, entrenado con el algoritmo de GloVe con los datos Spanish Billion Word Corpus, aplicando el algoritmo de PCA. Con un tamaño total de 1.1 GB

Con el fin de comprender con más profundidad la implicación de mantener constantemente este modelo en memoria, se presenta en la ilustración 6 una representación gráfica detallada. Esta ilustración muestra específicamente el escenario en el que nuestro sistema se ejecuta en un ordenador con una capacidad de memoria de 8GB.

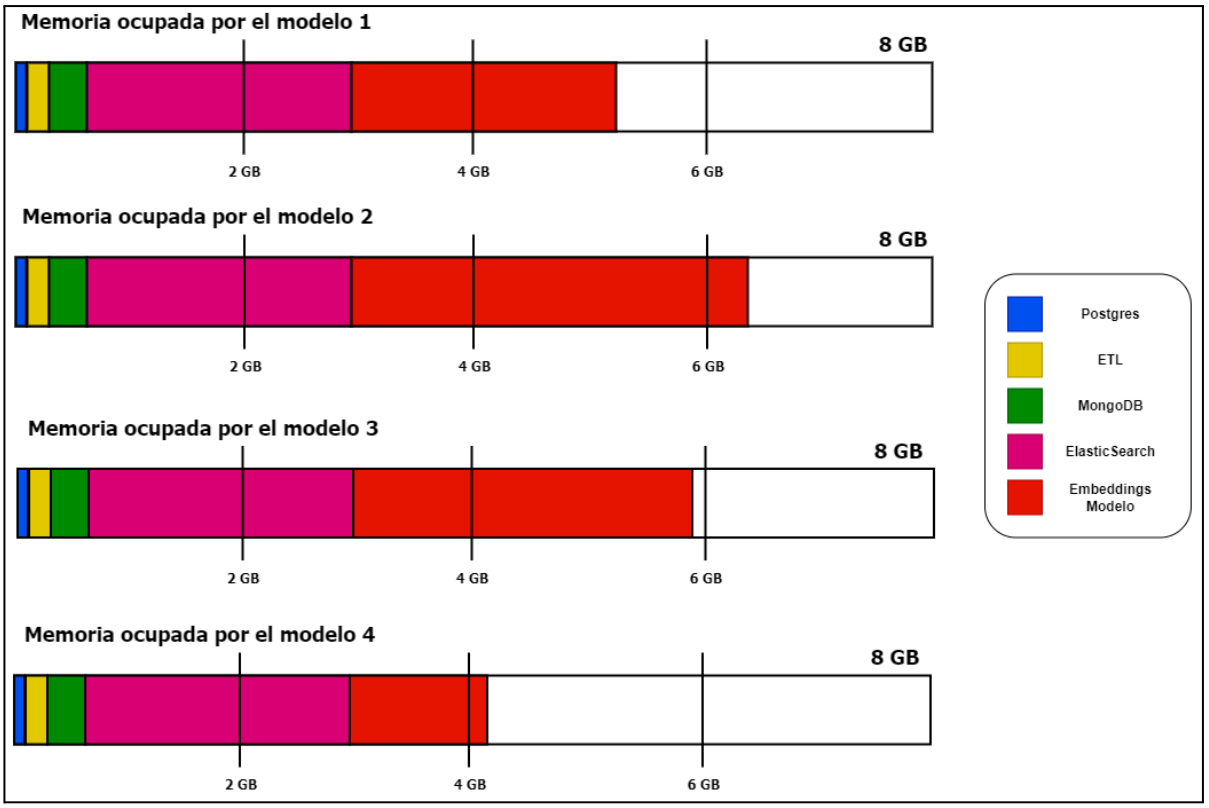

Ilustración 6: Diagrama de la memoria ocupada con los diferentes modelos generalistas.

Es comprensible que, en términos de gestión de memoria, sea más favorable utilizar el modelo que ha sido procesado mediante el algoritmo PCA. Dado que el algoritmo PCA reduce la dimensionalidad de los vectores y, por ende, el tamaño del modelo, resulta más eficiente en cuanto a la ocupación de memoria.

En este momento, es hora de realizar una evaluación para determinar cuál modelo ofrece los mejores resultados en las consultas. Sin embargo, antes de proceder con dicha evaluación, se debe llevar a cabo el proceso de almacenamiento e indexación de varias noticias para realizar pruebas adecuadas. Con este propósito, se han recopilado y almacenado todas las noticias publicadas a lo largo del año provenientes de todos los

portales. Como resultado, se cuenta con un conjunto total de aproximadamente 3500 noticias.

Para la indexación de los documentos se establecerán tres índices de embeddings, uno para el título, otro para la descripción y otro para el texto completo. El proceso de cálculo de los embeddings consistiría en tomar un título y generar una lista de palabras, excluyendo aquellas consideradas como stop-words[16]. A partir de esta lista de palabras, se calcularán los vectores correspondientes a cada una. Posteriormente, se obtendría el vector característico del título mediante el cálculo de la media aritmética de estos vectores. De manera análoga, se seguiría el mismo procedimiento para las descripciones y el cuerpo de cada noticia.
A cada vector se le asigna el identificador único correspondiente a la noticia en la base de datos, de manera que ElasticSearch pueda realizar su identificación de manera adecuada.
En un principio se valoró añadir las etiquetas en la lista de palabras del título pero, como se detalla en el anexo III, esto provoca que los embeddings se hacen más homogéneos, lo cual perjudica a las consultas.
Con el propósito de contar con un criterio adicional al momento de seleccionar el modelo que se utilizará, se ha tomado la decisión de medir el tiempo transcurrido entre la extracción y la indexación de las noticias para cada modelo, como se muestra en la ilustración 7. Esta métrica proporcionará información valiosa sobre el rendimiento y la eficiencia de cada modelo en términos de procesamiento de datos.

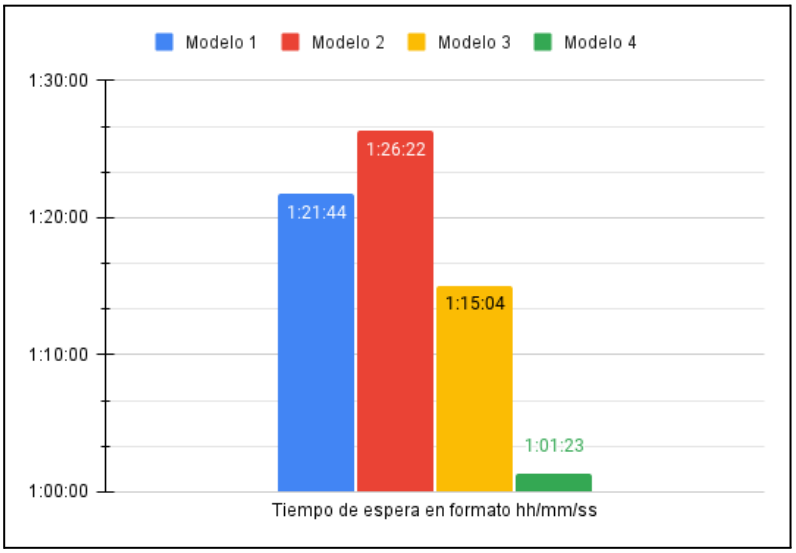

Ilustración 7: Gráfica con los tiempos de ejecución para cada modelo generalista.

Como es comprensible, el modelo entrenado con PCA es el que requiere menos tiempo para completar el proceso, aunque la diferencia es mínima. No obstante, es importante tener en cuenta que este caso de prueba es extremo, ya que se han extraído todas las noticias del año, lo cual equivale a aproximadamente cinco meses en la actualidad de datos.

[16] Stopword es una palabra común y no relevante que se filtra o elimina del texto durante el procesamiento del lenguaje natural para mejorar la eficiencia y precisión de los algoritmos de búsqueda y análisis de texto.

Sin embargo, en un escenario operativo normal, el sistema realiza extracciones semanalmente como máximo, lo que implica que los resultados obtenidos en esta situación extrema pueden no ser necesariamente relevantes para el rendimiento real del sistema en condiciones normales de uso.

Los resultados obtenidos para estos modelos no cumplieron con las expectativas establecidas, como se observa más en detalle en el anexo II, ya que en situaciones en las que no se encontraban documentos exactos que coincidieran con la consulta realizada, se obtuvieron resultados que no guardaban ninguna relación con el tema de la consulta. Se planteó la hipótesis de que los modelos utilizados eran tan generales que los documentos indexados presentaban vectores considerablemente homogéneos. Esta homogeneidad de los vectores implicaba una falta de distinción y diferenciación adecuadas entre los documentos, lo cual afectaba negativamente la capacidad del sistema para recuperar resultados relevantes y precisos en función de las consultas realizadas. Aparte de eliminar las etiquetas del título, se buscó un modelo de embeddings específico para el tema de la agricultura.

3.3.2 Modelos de embeddings especializados

Al realizar una búsqueda enfocada en modelos de embeddings especializados en el ámbito de la agricultura, se descubrió la existencia de un modelo BERT[17] entrenado con una amplia colección de 6,5 millones de textos relacionados con dicho dominio, a partir de ahora se denominará modelo 5. Este modelo se encontraba disponible en una página web[18] que permitía realizar pruebas directas utilizando el modelo. Tras llevar a cabo diversas pruebas, se constató que el modelo ofrecía los resultados deseados, cumpliendo con nuestras expectativas en cuanto a la relevancia y precisión de las respuestas obtenidas. Además, este modelo presenta la ventaja de poder recibir la frase completa como entrada, sin la necesidad de traducir palabra por palabra y calcular posteriormente la media aritmética, como se realizaba anteriormente con los modelos generalistas. Al recibir la frase completa, el modelo puede capturar mejor el contexto y la semántica global de la consulta, lo cual se traduce en resultados más precisos y relevantes.

Sin embargo, surgió un inconveniente: el modelo estaba en inglés, lo que implicaba la necesidad de traducir el texto antes de transformarlo en un vector. Después de considerar diversas opciones de tipo API, se tomó la decisión de utilizar la API de Deepl[19] para llevar a cabo la traducción requerida. Esta elección se encuentra detallada y justificada en el anexo I. Como alternativa, también se evaluó la posibilidad de utilizar una biblioteca de Python especializada en traducción basada en Deep Learning[20].

[17] BERT (Bidirectional Encoder Representations from Transformers) es un modelo de lenguaje basado en la arquitectura Transformer que captura el contexto y la semántica de las palabras en un texto. https://huggingface.co/docs/transformers/model_doc/bert

[18] https://huggingface.co/recobo/agriculture-bert-uncased

[19] DeepL es un servicio de traducción automática neuronal desarrollado por DeepL GmbH.

[20] Deep Learning es una rama del aprendizaje automático que utiliza redes neuronales artificiales profundas para aprender y extraer características de los datos. Permite el procesamiento de información y la toma de decisiones a partir de grandes conjuntos de datos de manera automática.

Además, como se observa en la ilustración 8, este modelo en comparación con los modelos previos requiere una menor cantidad de memoria, a excepción de cuando se emplea la biblioteca de traducción de Python, en cuyo caso su tamaño es similar al del modelo generalista más pequeño.

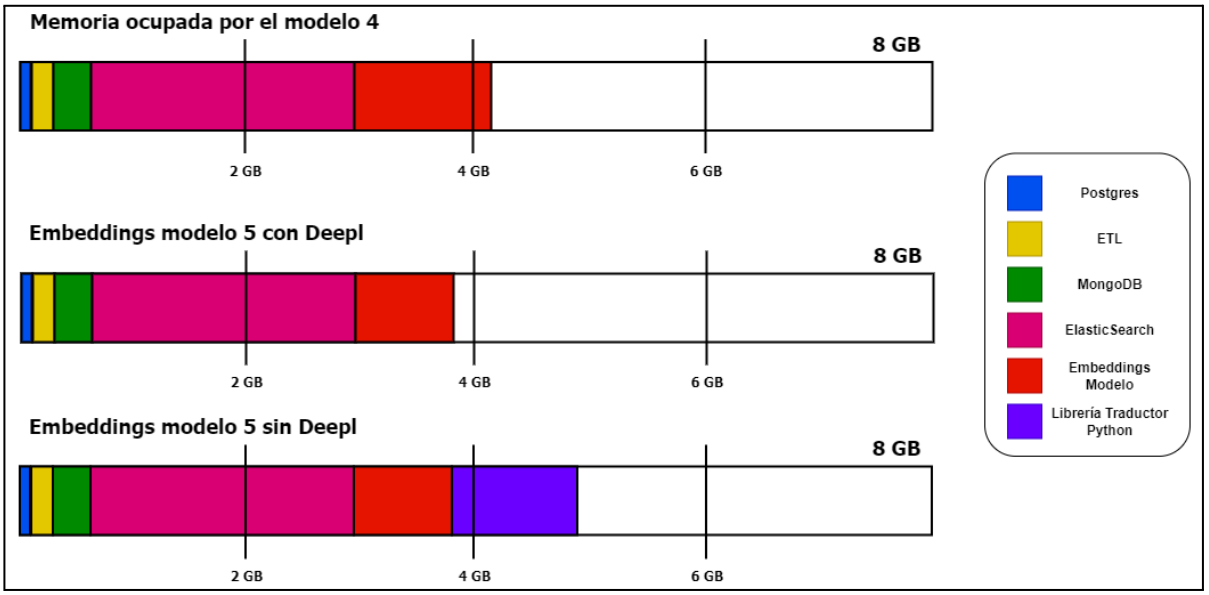

Ilustración 8: Diagrama de la memoria ocupada con los diferentes modelos especializados junto con el modelo generalista que tiene menor consumo de memoria.

Al igual que se realizó con los modelos de enfoque general, se llevó a cabo una medición del tiempo transcurrido entre la extracción y la indexación de las noticias para cada modelo, tal como se ilustra en el gráfico 9. Los resultados muestran claramente que el modelo que utiliza la API de Deepl es significativamente más rápido en comparación con los demás modelos evaluados. Por otro lado, se observa que el modelo que hace uso de la biblioteca de traducción de Python es notablemente más lento en comparación con los demás enfoques considerados.

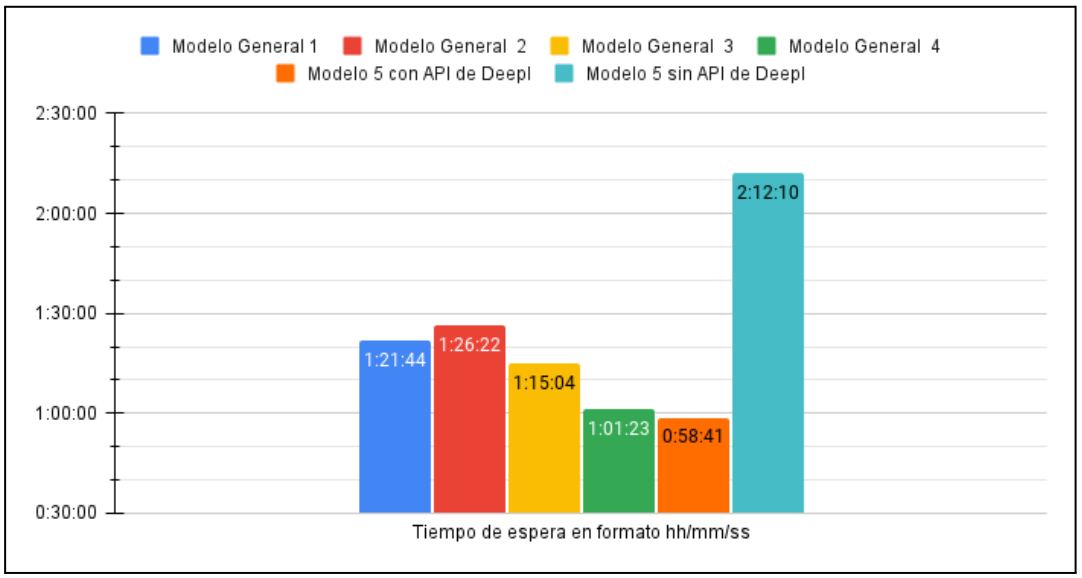

Ilustración 9: Gráfica con los tiempos de ejecución de todos los modelos.

Una vez analizados los resultados de rendimiento de nuestro sistema, procede examinar los resultados de las búsquedas efectuadas en el mismo. Con este propósito, se ha tomado la decisión de realizar un total de nueve consultas diferentes. Para cada una de ellas, se han calculado los parámetros de precisión, recall y k-precision correspondientes a cada modelo empleado. Estos cálculos se encuentran detallados en el anexo II, brindando una explicación más exhaustiva.
Una vez recopilados todos los resultados para cada consulta y modelo, se realizó un promedio de dichos parámetros con el fin de obtener una visión global de los resultados del sistema. Esta síntesis de los resultados se puede apreciar en la ilustración 10, que proporciona una visión consolidada de la eficacia del sistema en su conjunto.

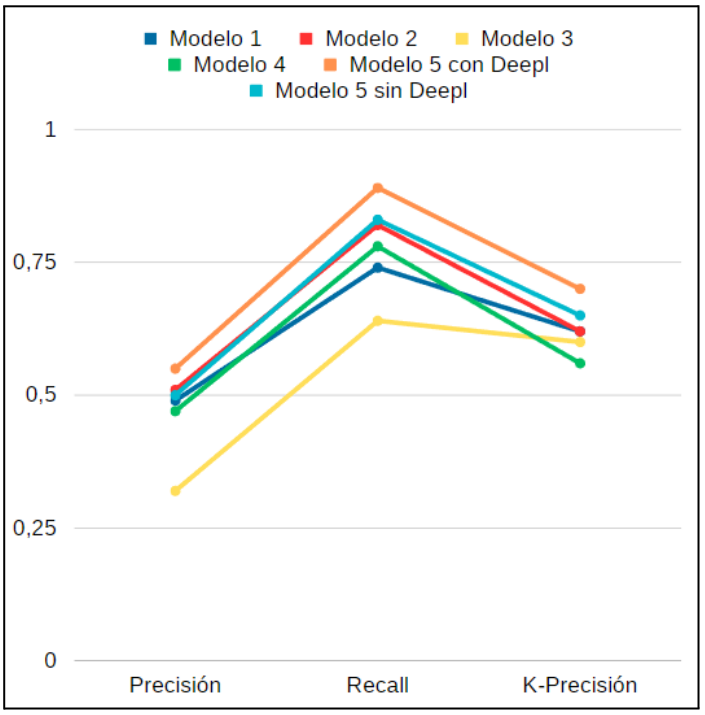

Ilustración 10: Gráfica con los datos de precisión, recall y k-precisión para cada modelo.

En general, los resultados obtenidos para estos modelos cumplieron las expectativas establecidas, ya que demostraron una capacidad para proporcionar resultados relevantes incluso en situaciones en las que no se encontraban documentos exactos que coincidieran con la consulta realizada.

De acuerdo con la información presentada en la ilustración 10, se puede apreciar que el modelo que ha exhibido los resultados más destacados es el modelo 5, conocido como el modelo BERT con la API Deepl para las traducciones. Es importante resaltar que este modelo corresponde a la versión final con la que se ha desarrollado el sistema.

3.4 Revisor de noticias

Después de completar el desarrollo de la funcionalidad ETL del sistema y adquirir una comprensión sólida sobre el funcionamiento de Apache Airflow y sus DAGs, se procedió a implementar el componente de revisión de noticias. El propósito de este componente es llevar a cabo una revisión semanal de todas las noticias almacenadas en la base de datos, verificando la accesibilidad de los enlaces asociados a cada noticia. Dado que la revisión estaba planificada para que ocurriera semanalmente sin futuras modificaciones, se tomó la decisión de crear un DAG específico para el revisor.

3.4.1 Control de accesibilidad

Para este componente lo principal fue desarrollar una función que permitiera extraer todos los enlaces que hubiera en la base de datos, para cada enlace se mandaba una petición HTTP de tipo GET, si el response de la petición devuelve un código de respuesta válido entonces la noticia seguía accesible, en caso contrario, habría que anotar un intento fallido para ese enlace.

Con el fin de mantener un registro de los intentos fallidos de los enlaces inaccesibles, se tomó la decisión de crear una variable que almacena una lista. Esta lista contiene el conjunto completo de enlaces inaccesibles, junto con el número de intentos asociado a cada uno. Además, esta variable también incluiría el número máximo permitido de intentos fallidos. De esta manera, cada vez que se incremente el contador de un enlace, se realizará automáticamente una comprobación para verificar si se ha alcanzado el número máximo de intentos. En caso afirmativo, la noticia correspondiente será eliminada tanto de la base de datos como del indexador, sin dejar ningún rastro de su existencia en el sistema.

Se consideró como otra opción la posibilidad de agregar directamente esa información en el documento de cada noticia, mediante la inclusión de un campo adicional. Sin embargo, se descartó esta opción con el objetivo de mantener la separación entre la lógica del proceso ETL y la lógica del revisor.

3.4.1 Control de la actualización del contenido

Durante el transcurso del desarrollo de este componente, surgió la idea de realizar una revisión adicional para verificar que las noticias no solo siguieran siendo accesibles, sino también que su contenido no hubiera experimentado cambios. Se observó que las noticias tienden a sufrir modificaciones debido a diversos motivos, como la obtención de nueva información relevante, la corrección de errores o imprecisiones en el contenido original, la actualización de datos estadísticos o la inclusión de eventos ocurridos después de la publicación inicial. Por lo tanto, se consideró importante incluir esta verificación para garantizar la integridad y actualidad de las noticias.

Con el objetivo de lograr esto, se implementó una funcionalidad adicional que consiste en obtener la fecha de publicación una vez que se haya verificado la accesibilidad del enlace. Esto permitiría comparar dicha fecha con la registrada en la base de datos, con el fin de determinar si la noticia ha sido actualizada. En caso de que las fechas difieran, se procederá a realizar nuevamente los procesos de ETL de manera específica para dicha noticia. Esto implica la actualización de la información en la base de datos y la recalculación de los embeddings en el indexador.

3.5 API

Una vez finalizada la implementación principal del backend de nuestro sistema, y habiendo establecido claramente nuestra estrategia de indexación de noticias, llegó el momento de desarrollar la API que brindará acceso a nuestros servicios desde el exterior. Esta API debe ser capaz de acceder a la capa de indexación para realizar consultas en el índice, así como a la capa de almacenamiento de las noticias para recuperar la información más relevante para el usuario.

Para la implementación de esta API, se utilizó el framework web de Python, Flask, el cual permite crear fácilmente una REST API[21] que proporciona un punto de acceso a las funcionalidades del sistema. Gracias a la simplicidad de su sintaxis, es posible lograr esta configuración en unas pocas líneas de código.

Era tiempo de desarrollar el principal servicio de la API que consiste en la búsqueda de noticias por palabras clave. Para desarrollar esto se configuró la API para tener una endpoint que permitiera peticiones HTTP de tipo GET que como parámetro se enviase las palabras clave de la consulta. Con ese parámetro se calculaba su vector de embeddings, similar a lo explicado en la sección 3.3.

Este enfoque de embedding ofrece la capacidad de realizar diversos tipos de consultas (anexo I); sin embargo, tras un análisis detenido, se determinó que el método más adecuado para nuestro caso específico es la búsqueda basada en la similitud del coseno[22] entre vectores. Esta elección se fundamenta en que este tipo de búsqueda es especialmente efectiva para tareas de recuperación de información en las que se busca encontrar documentos similares en función de su contenido semántico. En nuestro contexto, al buscar noticias relevantes para el usuario, resulta esencial considerar tanto la proximidad léxica como el significado contextual de los textos.

Conforme se detalla en la sección 3.3, al realizar una consulta en ElasticSearch, se obtendrá una lista de identificadores de documentos ordenados de manera descendente según su relevancia (Es posible especificar el número máximo de documentos que devolverá la consulta). A cada identificador se le asigna un valor denominado "score" un número real que representa el grado de relación entre la consulta realizada y el documento asociado a dicho identificador. Esto proporciona una medida cuantitativa de la similitud de cada documento en relación con la consulta ejecutada. Un valor más alto indica una mayor similitud y relevancia entre la consulta y el documento.

Mediante una serie de pruebas (anexo II), se vio que el valor referente para nuestro sistema será de 1.68 , a partir de este valor los documentos los podemos considerar como relevantes, para valores inferiores los consideraremos no relevantes.

[21] API REST es un conjunto de reglas y convenciones que permiten a los sistemas informáticos comunicarse entre sí a través de internet utilizando solicitudes y respuestas basadas en los protocolos HTTP y estándares web.

[22] La búsqueda basada en la similitud del coseno es un enfoque que compara la similitud entre dos vectores utilizando la medida del coseno del ángulo entre ellos. Es comúnmente utilizado para recuperar elementos similares en grandes conjuntos de datos, basándose en la proximidad de sus representaciones vectoriales en un espacio multidimensional.

La búsqueda tiene las siguientes etapas. En primer lugar, se llevaría a cabo una búsqueda relacionando la consulta con los títulos de las noticias, descartando aquellas que obtuvieran un puntaje inferior al umbral establecido previamente. En caso de que el número de noticias resultantes fuera insuficiente, se repetiría el proceso de búsqueda, esta vez solo considerando las descripciones asociadas a cada noticia. Nuevamente se aplicaría el filtrado y se agregarían con menor prioridad estas noticias junto con las obtenidas en la primera búsqueda. Por último, si aún no se alcanzaba el número mínimo de noticias, se repetiría el proceso una vez más, pero esta vez utilizando el cuerpo de las noticias.

Título → Descripción→ Cuerpo de la noticia

Este enfoque escalonado permitía refinar los resultados de la búsqueda en función de la relevancia y la cantidad de noticias obtenidas en cada etapa. Al comenzar con los títulos, se esperaba obtener una selección inicial de noticias que fueran altamente relevantes para la consulta. Luego, al ampliar la búsqueda a las descripciones, se considerarían noticias adicionales que pudieran complementar la información requerida. Finalmente, si aún fuera necesario, se exploraría el texto completo de las noticias para abarcar la mayor cantidad posible de contenido.

Por último, una vez obtenidos los identificadores de las noticias más relevantes a través del proceso de búsqueda, se procedía a recuperar de la base de datos toda la información asociada a dichas noticias. Esta información se transformaba posteriormente al formato JSON, con el fin de proporcionar una respuesta adecuada a la solicitud realizada mediante el protocolo HTTP. El proceso de recuperación de datos de la base de datos garantiza la obtención completa de la información necesaria para cada noticia, incluyendo detalles como el título, la descripción, el contenido y otros atributos relevantes. Al transformar esta información al formato JSON, se lograba una estructura estandarizada y legible, lo que facilitaba la transferencia de datos entre el servidor y el cliente a través de la respuesta HTTP.

3.6 Interfaz web

Por último, nos resta abordar la implementación de la interfaz web. Durante la evaluación de opciones para el desarrollo web, se consideraron diversos frameworks. Sin embargo, debido a la naturaleza de nuestra aplicación, la cual consta únicamente de una página con una barra de búsqueda en la que el usuario introduciría la consulta y sus resultados correspondientes, se determinó que no se requería un framework complejo.

Por consiguiente, se optó por utilizar una combinación de tecnologías estándar como HTML, CSS y Javascript vanilla (puro). Estas tecnologías ampliamente utilizadas y estandarizadas en el desarrollo web permiten crear una interfaz clara y funcional sin agregar una capa adicional de complejidad mediante el uso de frameworks más avanzados.

La elección de HTML como lenguaje de marcado, CSS para el diseño y estilos, y Javascript vanilla para la interactividad de la página garantiza la compatibilidad con una amplia gama de navegadores y dispositivos, y simplifica el proceso de desarrollo y mantenimiento. Las solicitudes a la API se realizan mediante el uso de Javascript. Al utilizar Javascript nos permite la actualización dinámica de los resultados y la manipulación de la información obtenida de la API. Además, con su soporte nativo para operaciones de red, es posible

realizar solicitudes HTTP de manera eficiente y obtener respuestas en formato JSON de la API, como se mencionó en la sección 3.5.

3.6.1 Diseño de la interfaz

Nuestro objetivo era diseñar una interfaz web que fuera sencilla y que proporcionase la información necesaria para cada noticia presentada, como se ve en la ilustración 11. Por lo tanto, resultaba fundamental mostrar el título de la noticia, junto con un enlace que permitiera redirigir fácilmente al contenido completo de la misma. Opcionalmente, se consideró incluir la fecha de publicación como información adicional para el usuario.

Ilustración 11: Interfaz web final.

La inclusión del título de la noticia garantiza que los usuarios tuvieran una vista rápida y concisa del tema principal de cada artículo. Esto permitía captar su atención y facilitaba la identificación de noticias relevantes en el contexto de su búsqueda o intereses específicos.

Además, proporcionar un enlace directo a la noticia permitía a los usuarios acceder rápidamente al contenido completo, lo cual resultaba crucial para aquellos que desearan explorar en detalle una noticia en particular. La posibilidad de mostrar opcionalmente la fecha de publicación brindaba información adicional para aquellos usuarios interesados en la actualidad de las noticias. Esto les permitía evaluar la frescura y relevancia temporal de los artículos presentados.

Finalmente, se implementó una funcionalidad de paginación que permite al usuario explorar a través de las flechas inferiores de la tabla, facilitando la visualización de un mayor número de resultados recomendados por el sistema.

4 CONCLUSIONES Y TRABAJO FUTURO

4.1 Conocimientos adquiridos

A lo largo de este Trabajo de Fin de Grado se ha encontrado una solución, aunque en ocasiones más efectiva que en otras, para abordar los problemas encontrados. Ha sido un viaje de descubrimiento sobre cómo iniciar y desarrollar un proyecto de tamaño mediano-grande. que sea real y persuasivo en términos de funcionalidad, escalabilidad y mantenibilidad a largo plazo. Esto implica la habilidad de diseñar de manera compleja, llevando a cabo un análisis inicial del dominio, y esforzándose por hacer el sistema lo más modular y sencillo posible.

Además, se ha adquirido un conocimiento profundo y se ha desarrollado habilidades en el uso de web scraping como un método eficaz de extracción de información web. Estos conocimientos adquiridos tienen una gran potencial para ser aplicados en proyectos futuros, ya que permite obtener información relevante de fuentes en línea de manera automatizada. Junto a eso, se ha adquirido un conocimiento sobre la complejidad de los procedimientos inherentes a un ETL. En particular, se ha reconocido que los datos extraídos de diferentes fuentes no siempre son homogéneos y requieren de una adaptación específica.

Sobre todo se han adquirido conocimientos sólidos sobre las diversas formas de indexación de documentos y los desafíos asociados a la creación de un indexador y buscador eficaz en todos los escenarios. Esto implica diseñar e implementar algoritmos y técnicas que permitan indexar y buscar eficientemente en grandes volúmenes de datos, garantizando una recuperación rápida y precisa de la información solicitada.

Por último, se ha aprendido a diseñar y crear una interfaz web de usuario atractiva, intuitiva y funcional que permita a los usuarios acceder y utilizar de manera efectiva las funcionalidades del sistema. El desarrollo frontend también ha implicado el conocimiento de prácticas de diseño *responsive*, que permiten que la aplicación se adapte y se visualice correctamente en diferentes dispositivos y tamaños de pantalla.

Además, se ha aprendido a virtualizar un sistema utilizando Docker, esto ha permitido crear un entorno de desarrollo y despliegue consistente y reproducible, independientemente del sistema operativo subyacente.

4.2 Trabajo futuro

Una vez realizada la primera versión de este sistema, que se centra en este Trabajo de final de Grado, tenemos que añadir muchos más portales de noticias, debido a que es un paso crucial en la evolución del sistema, con el objetivo de aumentar la cobertura y la diversidad de las fuentes de información permitiendo así enriquecer significativamente la base de datos del sistema, proporcionando una amplia variedad de perspectivas e información relevante para el ámbito agrícola.

En la capa de extracción, se ha identificado una posible mejora consistente en la implementación de una clase o módulo padre que contenga las funcionalidades básicas de extracción. De esta manera, a medida que se añadan nuevos portales web al sistema, se puedan crear clases o módulos hijos que hereden las funciones de la clase padre, pero que estén adaptados a la estructura y características específicas de cada web en particular. Esta estrategia permitirá mejorar la eficiencia y la escalabilidad del sistema, facilitando la integración de nuevos portales web de noticias en el futuro.

Además, es interesante echar un vistazo a la mejora del indexador y el modelo de embeddings debido a que ha sido la capa que mayores problemas ha dado en el desarrollo. Se puede plantear crear un modelo propio de embeddings mediante técnicas de aprendizaje automático a partir de un conjunto amplio de noticias de los portales que recoge nuestro sistema que proporcionará mejores resultados a las búsquedas.

En otro caso, se puede plantear el incorporar técnicas de procesamiento de lenguaje natural (NLP[23]) que permita realizar tareas de identificación de entidades relevantes (por ejemplo, nombres de cultivos, plagas o condiciones climáticas) y la clasificación automática de noticias según su temática o relevancia.

4.3 Conclusiones

En el presente trabajo, se ha abordado el diseño y desarrollo de un buscador y clasificador enfocado en la recopilación, transformación y carga de datos relacionados con noticias de agricultura. Este proyecto ha buscado abordar la necesidad de obtener información relevante y actualizada sobre el sector agrícola, con el fin de brindar apoyo a los profesionales y expertos en el campo de la agricultura.

Durante el proceso de diseño, se ha priorizado la simplicidad y modularidad de la arquitectura. Se han identificado y evaluado cuidadosamente los componentes esenciales necesarios para llevar a cabo las tareas. Esto ha permitido desarrollar una estructura flexible y escalable, capaz de adaptarse a diferentes fuentes de datos y requisitos específicos del dominio agrícola.

Durante la fase de extracción, se han identificado y analizado las diversas fuentes de datos disponibles, incluyendo portales web especializados, medios de comunicación y organismos gubernamentales, entre otros. La selección de estas fuentes se ha basado en su relevancia y fiabilidad para asegurar la calidad de la información recopilada. La etapa de transformación ha sido crucial para garantizar la homogeneización y estructuración de los datos extraídos. El proceso de carga ha permitido la inserción de los datos transformados en una base de datos especialmente para este propósito, donde se encuentran almacenados de manera organizada y accesible.

[23] Procesamiento del Lenguaje Natural (NLP) es una rama de la inteligencia artificial que se ocupa de la interacción entre las computadoras y el lenguaje humano, permitiendo a las máquinas comprender, analizar y generar texto de manera similar a como lo hacen los humanos. aws.amazon.com/es/what-is/nlp/

La implementación del indexador y buscador de noticias basados en embeddings ha implicado el uso de técnicas avanzadas de aprendizaje automático y procesamiento del lenguaje natural. A diferencia de los buscadores convencionales que se basan en la coincidencia de palabras clave en texto plano, esta solución proporciona consultas más relevantes y precisas. El enfoque de utilizar embeddings ha demostrado ser altamente efectivo para mejorar la calidad de las búsquedas en el sistema.

En cuanto a los resultados generales del sistema, se ha logrado desarrollar un sistema funcional y escalable que ha demostrado ser capaz de recopilar y transformar grandes volúmenes de datos relacionados con noticias de agricultura. La calidad y precisión de la información procesada han sido satisfactorias, lo que ha validado la eficacia del sistema implementado.

Además, se ha establecido una interfaz de usuario intuitiva y amigable, que permite a los usuarios realizar consultas y visualizar los datos de manera fácil y rápida. Esta interfaz ha sido diseñada teniendo en cuenta las necesidades y requerimientos de los profesionales del sector agrícola, brindando así una herramienta útil y eficiente.

LECTURAS RECOMENDADAS

¿Qué es un ETL?. (s.f.). Obtenido de AWS. https://aws.amazon.com/es/what-is/etl/

Inmon, W. H. (2002). Building the Data Warehouse. Wiley.

Carrillo, M., García-García, E., & Fernández, J. (2017). Análisis de datos agrícolas mediante técnicas de minería de textos. Revista de Investigación en Educación, Ciencia y Tecnología, Vol. 2, No. 2, pp. 45-59.

¿Qué es ElasticSearch?. (s.f.). Obtenido de Elastic.
https://www.elastic.co/es/what-is/elasticsearch

¿Qué son los word embeddings y para qué sirven? (s.f.).
https://blogs.iadb.org/conocimiento-abierto/es/que-son-los-word-embeddings/

Big Data. (s.f.). Obtenido de Oracle. https://www.oracle.com/es/big-data/what-is-big-data/

Kong, X., Song, X., & Li, S. (2019). Agricultural news classification based on a deep learning framework. Journal of Physics: Conference Series, Vol. 1343, No. 3.

Arquitectura de almacén de datos: tipos, componentes y conceptos. (s.f.). Obtenido de Astera. https://www.astera.com/es/knowledge-center/data-warehouse-architecture/

Comparativa de las herramientas ETL más usadas en la empresa. (s.f.). Obtenido de AprenderBigData. https://aprenderbigdata.com/herramientas-etl/

Cómo desplegar NLP: Incrustaciones de texto y búsqueda de vectores. (20 de mayo de 2022). Obtenido de Elastic.
https://www.elastic.co/es/blog/how-to-deploy-nlp-text-embeddings-and-vector-search

Qué es la arquitectura en capas, ventajas y ejemplos. (s.f.). Obtenido de HubSpot:
https://blog.hubspot.es/website/que-es-arquitectura-en-capas

Documentación de Apache Airflow. (s.f.). Obtenido de Apache Airflow.
https://airflow.apache.org/docs/apache-airflow/stable/index.html

What Is A Data Lake?. (s.f.). Obtenido de MongoDB.
https://www.mongodb.com/databases/data-lake-explained

Kimball, R., Ross, M., Thornthwaite, W., Mundy, J., & Becker, B. (2011). The Data Warehouse Lifecycle Toolkit: Expert Methods for Designing, Developing, and Deploying Data Warehouses (2nd ed.). Wiley.

ANEXO I. UN POCO MÁS DEL PROTOTIPO

En este anexo se van a explicar de forma más detallada los aspectos más técnicos y centrados en el diseño del sistema.

Tecnologías más en profundidad

Como hemos contado en la sección 1.2 de la memoria, se han hecho uso de varias tecnologías, cada una con su propósito claro. Ahora, para cada una de las tecnologías que se han escogido, se explica el porqué de dicha elección sobre otras tecnologías:

- *Python* es un lenguaje de programación de alto nivel, interpretado, multiparadigma y de propósito general. Python se destaca por su sintaxis intuitiva y fácil de leer. Utiliza una estructura de indentación significativa en lugar de llaves o palabras clave para delimitar bloques de código, lo que mejora la legibilidad y reduce los errores de sintaxis. Esta claridad sintáctica facilita la comprensión y el mantenimiento del código, tanto para programadores principiantes como experimentados.Python cuenta con una biblioteca estándar extensa y poderosa que abarca una amplia gama de tareas, desde manipulación de cadenas y expresiones regulares hasta acceso a redes y procesamiento de datos. La biblioteca estándar permite a los desarrolladores ahorrar tiempo y esfuerzo al proporcionar soluciones predefinidas y de alto nivel para muchos problemas comunes.Python admite múltiples paradigmas de programación, incluyendo programación imperativa, orientada a objetos y funcional. Esto proporciona flexibilidad al programador para elegir el enfoque más adecuado según las necesidades del proyecto. Python también es ampliamente utilizado en campos como el análisis de datos, la inteligencia artificial y el desarrollo web, lo que demuestra su versatilidad y capacidad para adaptarse a diversos dominios.
- *Beautiful Soup* es una biblioteca de Python que se utiliza para extraer información de páginas web y analizar documentos HTML y XML. Proporciona herramientas para analizar y navegar la estructura del código fuente de una página web, buscar y extraer datos específicos, y manipular el contenido de manera conveniente. Beautiful Soup simplifica el proceso de análisis de páginas web al proporcionar métodos y funciones que permiten extraer información específica de manera fácil y eficiente. Puede analizar y navegar por la estructura del HTML/XML, buscar elementos por etiquetas, atributos o contenido, y extraer los datos necesarios para su posterior procesamiento. Muchas páginas web tienen una estructura jerárquica y anidada con múltiples elementos y etiquetas. Beautiful Soup permite acceder y manipular estas estructuras de manera intuitiva, brindando métodos como find, find_all y select que facilitan la navegación y búsqueda de elementos específicos en el código fuente. Beautiful Soup puede trabajar con diferentes analizadores HTML/XML, como html.parser, lxml, html5lib, entre otros. Esto permite adaptarse a las necesidades específicas del proyecto y utilizar el analizador más adecuado para obtener el mejor rendimiento y precisión en el análisis de las páginas web.
- *Pandas* es una biblioteca de Python ampliamente utilizada para el análisis y manipulación de datos. Proporciona estructuras de datos y herramientas de alto rendimiento que facilitan la limpieza, transformación, exploración y visualización de

datos, lo que la convierte en una herramienta fundamental en el ámbito del análisis de datos. Pandas ofrece una amplia gama de funciones y métodos para realizar tareas comunes de manipulación y limpieza de datos. Puede filtrar, ordenar, agregar, fusionar y transformar datos con facilidad. Además, Pandas proporciona herramientas para tratar valores faltantes, eliminar duplicados, cambiar tipos de datos y manejar errores en los datos, lo que agiliza el proceso de preparación de datos para el análisis. Pandas está diseñado para manejar grandes conjuntos de datos de manera eficiente. Utiliza estructuras de datos optimizadas y algoritmos de procesamiento vectorizados de NumPy, lo que mejora el rendimiento y reduce el tiempo de ejecución. Además, Pandas ofrece funciones para leer y escribir datos en diferentes formatos, como CSV, Excel, SQL y HDF5, lo que facilita la importación y exportación de datos.

- *Apache Airflow* es una plataforma de código abierto diseñada para programar, monitorear y orquestar flujos de trabajo (workflows) de datos. Proporciona una interfaz de usuario intuitiva y herramientas para definir, programar y ejecutar tareas de manera eficiente. Airflow es altamente escalable y flexible, lo que lo convierte en una opción popular para la automatización de flujos de trabajo en entornos de datos complejos. Airflow permite definir y programar flujos de trabajo como un conjunto de tareas interconectadas. Los flujos de trabajo se definen utilizando código Python, lo que proporciona flexibilidad y permite la expresión de la lógica empresarial compleja. Airflow utiliza una sintaxis declarativa y permite la programación basada en tiempo o eventos, lo que facilita la automatización de tareas repetitivas. Airflow proporciona un panel de control centralizado para administrar y monitorear flujos de trabajo. Permite visualizar el estado y el progreso de las tareas, así como ver el historial de ejecución y los registros de las tareas. Esto facilita la supervisión y solución de problemas en tiempo real, lo que ayuda a mantener el control sobre los flujos de trabajo y garantizar su ejecución exitosa. Airflow es altamente escalable y puede manejar flujos de trabajo complejos y de gran tamaño. Puede ejecutar tareas en paralelo y distribuir la carga de trabajo en múltiples nodos, lo que mejora la eficiencia y el rendimiento. Airflow también admite la programación de flujos de trabajo en clústeres de servidores, lo que permite la escalabilidad horizontal para manejar grandes volúmenes de datos y tareas concurrentes. Airflow se integra con una amplia gama de herramientas y servicios comúnmente utilizados en el ecosistema de datos. Puede interactuar con bases de datos, almacenamiento en la nube, sistemas de colas, herramientas de programación y otras tecnologías, lo que facilita la incorporación de Airflow en infraestructuras existentes.
- *MongoDB* es una base de datos NoSQL (No Relacional) que se caracteriza por su enfoque en almacenar y recuperar datos en forma de documentos JSON (JavaScript Object Notation). Fue diseñada para manejar grandes volúmenes de datos, ofreciendo flexibilidad y escalabilidad. A diferencia de las bases de datos relacionales tradicionales, MongoDB no utiliza tablas con filas y columnas. En su lugar, utiliza colecciones de documentos, donde cada documento es una estructura de datos flexible y autocontenida que puede variar en su forma y contenido. Cada documento se almacena en un formato similar a JSON, lo que permite una fácil representación y manipulación de datos complejos. MongoDB permite almacenar documentos con estructuras diferentes en una misma colección. Esto es especialmente útil en aplicaciones que manejan datos no estructurados o que tienen

requisitos cambiantes. No es necesario definir un esquema rígido de antemano, lo que facilita la evolución de la base de datos con el tiempo. MongoDB utiliza un modelo de almacenamiento en memoria caché y un sistema de indexación eficiente para acelerar las consultas y las operaciones de lectura y escritura. Además, al aprovechar la estructura de documentos JSON, se evita la necesidad de realizar un mapeo entre objetos en la aplicación y filas en la base de datos, lo que reduce la latencia y mejora el rendimiento. MongoDB utiliza un modelo de almacenamiento en memoria caché y un sistema de indexación eficiente para acelerar las consultas y las operaciones de lectura y escritura. Además, al aprovechar la estructura de documentos JSON, se evita la necesidad de realizar un mapeo entre objetos en la aplicación y filas en la base de datos, lo que reduce la latencia y mejora el rendimiento. MongoDB proporciona mecanismos de replicación automática que garantizan la disponibilidad continua de los datos incluso en caso de fallos de hardware o interrupciones. Los datos se pueden replicar en varios servidores en tiempo real, lo que brinda tolerancia a fallos y mayor confiabilidad.

- *PostgreSQL* es un sistema de gestión de bases de datos relacional de código abierto, altamente escalable y confiable. Es conocido por su robustez, flexibilidad y soporte de características avanzadas, lo que lo convierte en una opción popular para aplicaciones empresariales y de alto rendimiento. PostgreSQL es conocido por su confiabilidad y estabilidad. Es un sistema de base de datos robusto que ha demostrado ser capaz de manejar grandes volúmenes de datos y cargas de trabajo intensivas. Está diseñado para garantizar la integridad de los datos, con mecanismos de recuperación y detección de errores incorporados. PostgreSQL ofrece una amplia gama de características avanzadas que lo distinguen de otros sistemas de gestión de bases de datos. Soporta transacciones ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), lo que garantiza la integridad y consistencia de los datos. Además, proporciona soporte para consultas complejas, claves foráneas, disparadores (triggers), vistas materializadas, replicación y particionamiento, entre otras características.
- *Elasticsearch* es un motor de búsqueda y análisis de datos distribuido, de código abierto y altamente escalable. Está basado en el motor de búsqueda de texto completo Lucene y proporciona una solución eficiente para indexar, buscar y analizar grandes volúmenes de datos en tiempo real. Elasticsearch utiliza una estructura de índices invertidos y un modelo de vector de espacio para indexar y buscar datos de manera rápida y precisa. Permite realizar búsquedas de texto completo, búsqueda geoespacial, búsqueda por facetas y búsqueda de texto relevante. Esto hace que sea fácil y rápido encontrar información relevante en grandes volúmenes de datos. Elasticsearch está diseñado para ser altamente escalable y distribuido. Puede manejar grandes volúmenes de datos y escalar horizontalmente agregando más nodos al clúster. Además, Elasticsearch distribuye y replica automáticamente los datos a través de los nodos para proporcionar alta disponibilidad y tolerancia a fallos. Elasticsearch permite realizar análisis y agregaciones en tiempo real sobre los datos indexados. Proporciona una amplia gama de agregaciones estadísticas, como sumas, promedios, máximos, mínimos y conteos, así como también agregaciones avanzadas, como histogramas, términos y métricas compuestas. Esto facilita el análisis y la extracción de información significativa de los datos almacenados en Elasticsearch. Elasticsearch admite

diferentes tipos de datos y formatos, como texto, numéricos, geoespaciales y estructurados. Puede indexar datos estructurados y no estructurados de manera flexible y proporciona opciones de análisis de texto, como tokenización, filtrado y normalización, para mejorar la calidad de las búsquedas y la relevancia de los resultados. Elasticsearch proporciona herramientas para monitorizar y supervisar el rendimiento del clúster, los índices y las consultas. Permite realizar un seguimiento de la latencia, el uso de recursos y la carga de trabajo, lo que facilita la identificación de cuellos de botella y la optimización del rendimiento. Además, Elasticsearch permite configurar alertas para notificar sobre eventos o condiciones específicas, lo que ayuda a detectar problemas y tomar medidas proactivas.

En cuanto a la parte de *devOps,* contamos con Docker para lanzarlo, el cual, de momento, es más que suficiente para meterlo todo en contenedores de forma que se pueda desplegar en cualquier servidor de forma sencilla.

Diseño del sistema más detallado

En la sección 2 de la memoria se ha examinado y explicado el diseño del sistema en su conjunto. En esta sección se proporcionará una explicación más exhaustiva de cada capa que conforma el sistema, así como de la interacción entre los componentes del mismo.

Capas de ETL

Estas capas desempeñan la función de extraer información de los sitios web, transformarla en datos estructurados y almacenarla en la base de datos. Como se ha explicado previamente, el controlador de flujo del sistema lanza diariamente N procesos, donde N representa el número de portales admitidos por nuestro sistema. Cada uno de estos procesos invoca al módulo encargado de verificar si corresponde extraer noticias de su portal en particular, realizando esta verificación comparando la fecha actual con la fecha almacenada en la base de datos, junto con la frecuencia en días establecida para las tareas de ETL de dicho portal. Con base en esta información, se devuelve un valor booleano. Si el valor booleano es positivo, se procede a extraer las noticias desde el último día en que se realizó el trabajo hasta la fecha actual, transformándolas y almacenándolas en la base de datos.

Todas las operaciones que interactúan con la base de datos se llevan a cabo utilizando el mecanismo de exclusión mutua y transacciones. Esto se debe a la naturaleza concurrente del sistema, donde varios procesos se ejecutan simultáneamente y acceden al mismo recurso, es decir, la base de datos.

La exclusión mutua garantiza que solo un proceso pueda acceder y modificar la base de datos en un momento dado, evitando así posibles conflictos e inconsistencias en los datos. Esto se logra mediante el uso de bloqueos y semáforos para asegurar la integridad de las operaciones y evitar condiciones de carrera.

Por otro lado, el uso de transacciones proporciona una forma coherente y fiable de gestionar las operaciones en la base de datos. Las transacciones permiten agrupar un conjunto de operaciones relacionadas en una unidad lógica y asegurar que se ejecuten de manera atómica, es decir, todas o ninguna de las operaciones se llevan a cabo. Además, las transacciones ofrecen propiedades ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), lo que

garantiza que los cambios en la base de datos sean consistentes y duraderos incluso en caso de fallos del sistema.

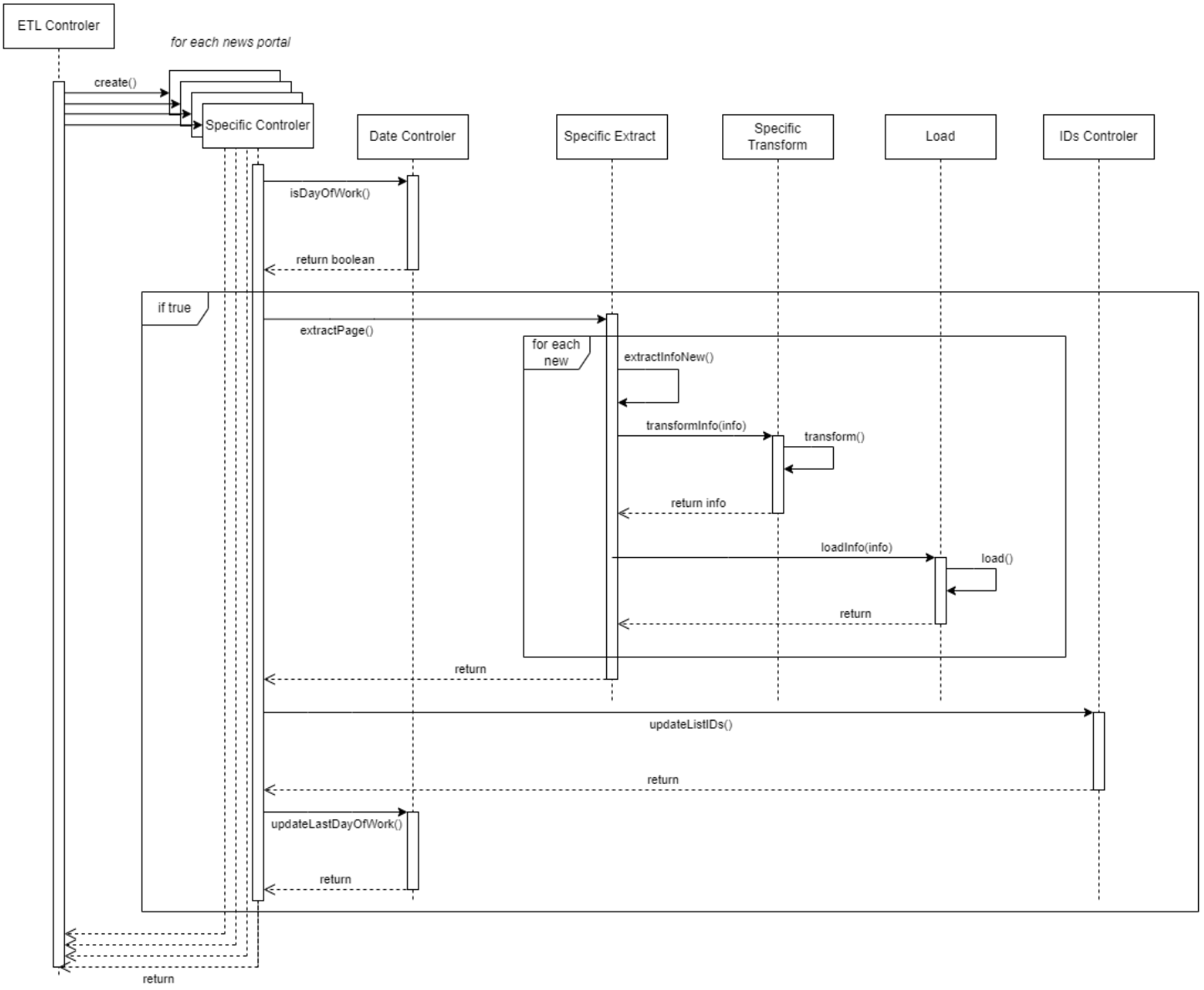

Ilustración 12: Diagrama de secuencia de los módulos que componen el ETL.

Una vez que la noticia se almacena en la base de datos, se extrae su identificador único, el cual se carga en la lista de identificadores correspondientes a las noticias que aún no han sido indexadas, tal y como se explicó en la sección 3.3, donde se tomó esta decisión. Una vez que se procesan todas las noticias mediante el módulo de fechas, se actualiza la última fecha de trabajo con la fecha actual. Esta etapa del sistema se puede visualizar de manera más clara en la ilustración 12 mediante un diagrama de secuencia.

Capa del indexador

Esta capa de indexación tiene la responsabilidad de indexar todas las noticias almacenadas en la base de datos que aún no han sido indexadas. Con el objetivo de optimizar los recursos, tal como se ha explicado previamente, el controlador de flujo del sistema realiza el proceso de indexación de forma semanal.

Cada semana, el indexador recupera la lista de identificadores de las noticias pendientes de indexar. En caso de que la lista contenga documentos, se carga en memoria el modelo de embeddings. Este proceso de carga tiene una duración de unos pocos minutos. Una vez cargado, se procede a iterar sobre cada identificador, extrayendo su contenido de la base de datos. Posteriormente, se transforma dicho contenido en un vector de embeddings y se almacena en el módulo de ElasticSearch. Esta etapa del sistema se puede visualizar de manera más clara en la ilustración 13 mediante un diagrama de secuencia.
Cuando la lista de identificadores se encuentra vacía, se actualiza la fecha actual como el último día en el que se ejecutó la indexación. Esto asegura que el sistema tenga un registro actualizado del momento en que se realizó la última indexación.

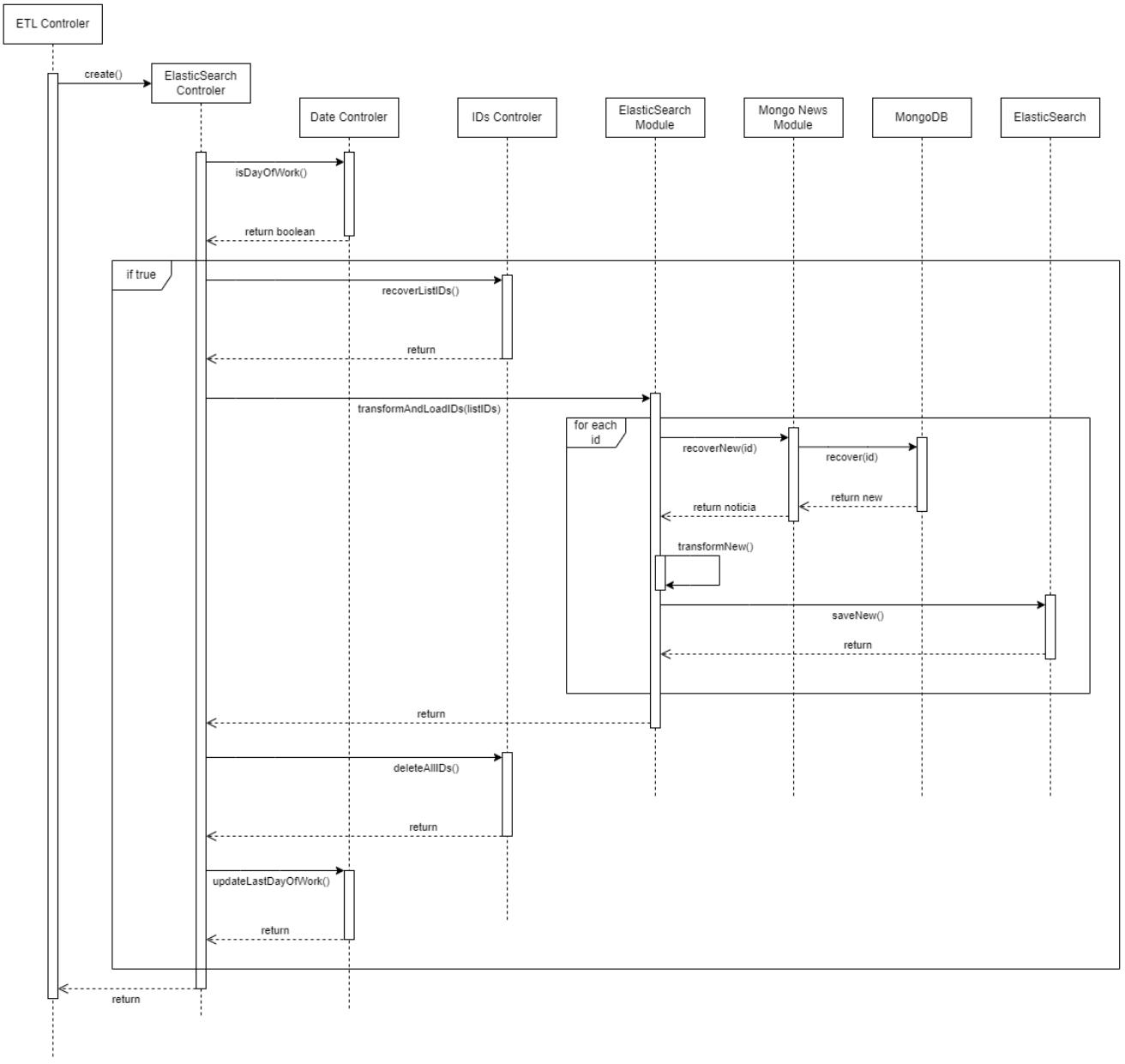

Ilustración 13: Diagrama de secuencia de los módulos que componen el indexador.

Capa del revisor

Esta capa cumple la función de supervisar la accesibilidad y la integridad del contenido de todas las noticias almacenadas en la base de datos. Como se ha mencionado previamente, el controlador de flujo del sistema lanza semanalmente N procesos, donde N representa el número de portales admitidos por nuestro sistema. Cada proceso se encarga de verificar la accesibilidad de los enlaces asociados a un portal específico.

Para llevar a cabo esta verificación, se recuperan de la base de datos todos los enlaces existentes asignados al portal en cuestión. Por cada enlace, se realiza una solicitud HTTP para comprobar su disponibilidad y se verifica el código de respuesta obtenido. Si el código de respuesta indica que el enlace es accesible, se procede a verificar si el contenido de la noticia ha cambiado comparando la fecha asociada en la base de datos con la fecha actual del contenido web. En caso de que sean diferentes, se actualiza el contenido mediante las funciones de Extracción, Transformación y Carga (ETL) correspondientes a esa noticia en particular. Si el enlace no es accesible, se registra un intento fallido para dicho enlace. Si el número de intentos asociados a la noticia supera un límite predefinido, entonces se elimina dicha noticia de la base de datos. Esta etapa del sistema se puede visualizar de manera más clara en la ilustración 14 mediante un diagrama de secuencia.

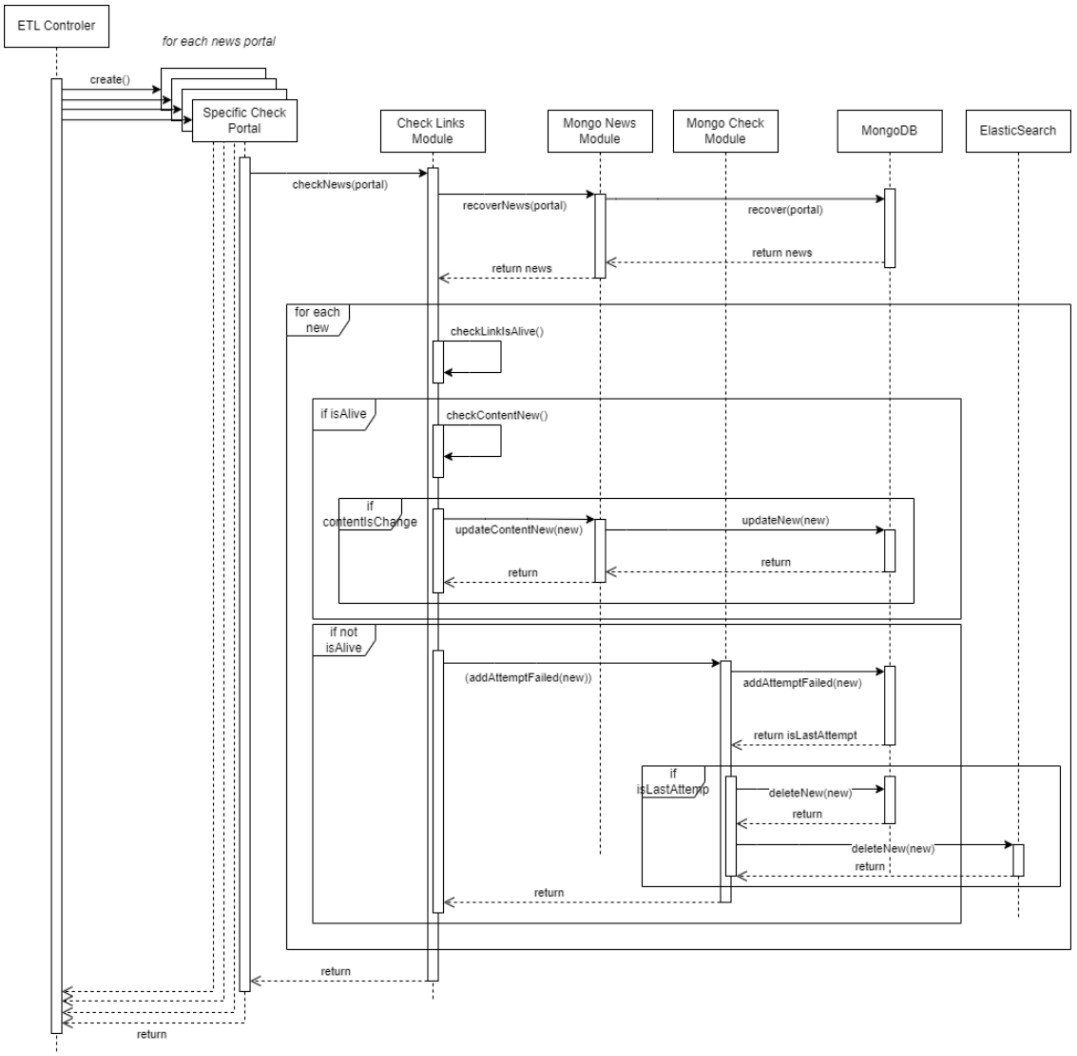

Ilustración 14: Diagrama de secuencia de los módulos que componen el revisor.

Estructura y organización del sistema a nivel de código

Es interesante ver como se ha estructurado y realzado el sistema a nivel del código. Se ha tratado de realizar un código limpio, modular, escalable y mantenible en el tiempo.

```
-
|-- README.md
|-- .gitignore
|-- model
|-- api
|     |-- api_endpoint.py
|     |-- api_functions.py
|-- dags
|     |-- genreal_dag.py
|     |-- revisor_enlaces_dag.py
|-- elastic_search
|     |-- elasticsearch_config.py
|     |-- elasticsearch_func.py
|     |-- elasticsearch_utils.py
|-- embedding
|     |-- embedding_transformer.py
|     |-- traductor.py
|-- extract
|     |-- extract_general.py
|     |-- extract_page1.py
|     |-- extract_page2.py
|     |-- extract_page3.py
|     |-- extract_page4.py
|     |-- extract_utils.py
|-- load
|     |-- load.py
|     |-- load_utils.py
|-- mongo
|     |-- mongo_fechas.py
|     |-- mongo_ids.py
|     |-- mongo_noticias.py
|     |-- mongo_revisor.py
|-- web
|     |-- css
|     |-- imagenes
|     |-- js
|     |-- index.html
|-- revisor_enlaces
|     |-- revisor_enlaces.py
|-- Dockerfile.app
|-- Dockerfile.mongo
|-- config_mongo.js
|-- docker-compose.yaml
|-- entrypoint.sh
|-- requirements.txt
-
```

Listado 1: Estructura de carpetas y ficheros.

El sistema se encuentra desplegable a través del archivo *docker-compose.yaml*, el cual despliega de manera integral los contenedores necesarios para su funcionamiento. En particular, el archivo *Dockerfile.app* asume un papel fundamental, ya que se encarga de crear el contenedor de Python y Apache Airflow. En este proceso, se copia el código del sistema, se instalan las dependencias necesarias (según las especificaciones del archivo requirements.txt) y se ejecutan las configuraciones iniciales (mediante el archivo entrypoints.sh).

Por otro lado, el archivo *Dockerfile.mongo* desempeña una función esencial al crear el contenedor de mongoDB. Además, se encarga de cargar los datos iniciales necesarios para el funcionamiento del sistema. Estos datos son extraídos del archivo *config_mongo.js*.

La combinación de estos elementos, junto con la orquestación proporcionada por el archivo *docker-compose.yaml*, constituye una solución completa y robusta que facilita la puesta en marcha y el mantenimiento eficiente del sistema en cuestión.

Diseño y técnicas sobre la indexación

Como se ha destacado previamente en las secciones anteriores, se ha optado por emplear la metodología de embeddings como estrategia de indexación en nuestro sistema. Esta elección se basa en los beneficios sustanciales que brindan, como una representación semántica más sólida, la capacidad de realizar búsquedas basadas en similitud y la mejora en la precisión y relevancia de los resultados obtenidos. Sin embargo, es relevante mencionar que se han considerado otras alternativas igualmente valiosas que merecen ser exploradas en profundidad:

- *Índices invertidos*: Los índices invertidos son una estructura de datos ampliamente utilizada en los motores de búsqueda. Almacenan información sobre las palabras clave y los documentos en los que aparecen. Esto permite una búsqueda eficiente de documentos a partir de palabras clave y facilita la recuperación de información relevante.
- *Índices de texto completo*: Estos índices se utilizan para indexar y buscar texto completo, donde cada palabra o término se almacena junto con su posición en el documento. Los índices de texto completo permiten búsquedas rápidas basadas en palabras clave y pueden incluir funciones avanzadas como la coincidencia de sinónimos y la relevancia de los términos de búsqueda.
- *Indexación basada en estructura*: Algunos sistemas de recuperación de información indexan la estructura de los documentos, como encabezados, secciones y etiquetas HTML. Esto permite búsquedas específicas en partes específicas de los documentos y facilita la recuperación de información estructurada.
- *Indexación temporal*: En los sistemas que manejan datos temporales, como noticias o publicaciones en redes sociales, la indexación temporal es útil. Los índices temporales permiten buscar documentos en función de su fecha o rango de fechas, lo que facilita la recuperación de información histórica.
- *Indexación por atributos*: Algunos sistemas indexan los documentos en función de atributos específicos, como autor, ubicación geográfica, etc. Esto permite realizar búsquedas más refinadas y precisas, centrándose en características específicas de los documentos.

- *Indexación por metadatos*: Los metadatos son datos descriptivos sobre los documentos, como título, autor, fecha de creación, etc. Al indexar los metadatos de los documentos, se puede realizar una búsqueda más rápida y precisa basada en criterios específicos.
- *Indexación por entidades nombradas*: En este enfoque, se identifican y se indexan entidades nombradas, como nombres de personas, organizaciones o lugares, presentes en los documentos. Esto permite búsquedas y filtrado basados en entidades específicas, lo que resulta útil para la búsqueda de noticias relacionadas con personas o empresas específicas.

Además de la metodología seleccionada para la indexación, se debía tomar una decisión con respecto a la metodología a emplear para las búsquedas. Tal y como se expuso anteriormente, se optó por utilizar la técnica de búsqueda por similitud del coseno, pero es relevante mencionar otras alternativas que se han considerado:

- *Búsqueda por vecinos más cercanos*: En esta técnica, se busca encontrar los documentos cuyos vectores están más cerca en el espacio vectorial al vector de consulta. Se utilizan algoritmos de búsqueda de vecinos más cercanos, como el K-NN (k-nearest neighbors), para identificar los documentos más similares al vector de consulta.
- *Búsqueda por aproximación*: Algunos métodos de búsqueda basados en embeddings utilizan técnicas de aproximación para reducir la complejidad computacional y acelerar el proceso de búsqueda. Estos métodos pueden incluir el uso de estructuras de datos como los árboles KD o el hashing para indexar y buscar los embeddings de manera más eficiente.
- *Búsqueda por combinación de características*: En esta técnica, se utilizan múltiples características o embeddings para realizar la búsqueda. Por ejemplo, se pueden utilizar embeddings de texto junto con embeddings de imágenes o audio para realizar una búsqueda multimodal que combine diferentes modalidades de información.

Motivos para la elección de la API Deepl

En el campo de la traducción automática, existen varias APIs disponibles para facilitar el proceso de traducción de textos. Sin embargo, en los últimos tiempos, la API de DeepL ha ganado popularidad y ha sido elegida por muchos desarrolladores y profesionales de la traducción sobre otras opciones disponibles. A continuación, analizaremos algunas razones clave por las que la API de DeepL se ha convertido en una elección preferida en comparación con otras APIs de traducción.

En primer lugar, la precisión y la calidad de las traducciones generadas por la API de DeepL han impresionado a muchos usuarios. Basada en la tecnología de traducción neuronal, DeepL ha logrado alcanzar niveles sorprendentes de precisión y fluidez en la traducción de textos. Sus algoritmos avanzados y el extenso entrenamiento con grandes volúmenes de datos han permitido a DeepL ofrecer traducciones que a menudo se consideran más naturales y contextuales en comparación con otras opciones.

Otro aspecto destacado de la API de DeepL es su enfoque en la privacidad y la seguridad de los datos. Muchos usuarios aprecian la política de privacidad de DeepL, que se enfoca en proteger la confidencialidad de los textos traducidos.

Además, DeepL ha demostrado ser una opción escalable para proyectos que involucran grandes volúmenes de traducciones. Su capacidad para manejar grandes cargas de trabajo y su tiempo de respuesta rápido han sido puntos destacados para muchos usuarios en el ámbito de la informática, donde la eficiencia y la velocidad son aspectos críticos.

ANEXO II. RESULTADOS OBTENIDOS

Como ya se ha mencionado en la sección 3.3 los modelos de embeddings que se quieren estudiar son los siguientes:

- **Modelo 1**: Modelo de 985.667 vectores con 300 componentes cada uno, entrenado con el algoritmo de FastText con datos de la wikipedia en español. Con un tamaño total de 2.2 GB.
- **Modelo 2**: Modelo de 1.313.423 vectores con 300 componentes, entrenado con el algoritmo de GloVe de SBWC. Con un tamaño total de 3.4 GB
- **Modelo 3**: Modelo de 1.313.423 vectores con 300 componentes, entrenado con el algoritmo de FastText de SUC. Con un tamaño total de 2.9 GB
- **Modelo 4**: Modelo de 1.313.423 vectores con 100 componentes, entrenado con el algoritmo de GloVe de SBWC, aplicando el algoritmo de PCA. Con un tamaño total de 1.1 GB
- **Modelo 5 con API Deepl**: Un modelo BERT especializado en el tema sobre la agricultura, entrenado con una amplia colección de 6,5 millones de textos relacionados con dicho dominio. Se utiliza la API de Deepl para traducir los textos.
- **Modelo 5 con traductor de Python**: Un modelo BERT especializado en el tema sobre la agricultura, entrenado con una amplia colección de 6,5 millones de textos relacionados con dicho dominio. Se utiliza una biblioteca de Python especializada en traducción basada en Deep Learning.

Una vez se ha establecido con claridad el conjunto de modelos a emplear con el propósito de obtener resultados significativos, se procede a la etapa de selección del modelo óptimo. En este punto, se determinan las consultas que se llevarán a cabo utilizando los datos disponibles con el fin de evaluar exhaustivamente los modelos.

Durante este proceso, se realiza una cuidadosa elección de las consultas a establecer, como se observa en la tabla 1, considerando su relevancia y representatividad para abarcar de manera completa las características y variaciones presentes en los datos. Esta selección estratégica de consultas busca proporcionar una evaluación integral y precisa de los modelos en cuestión.

CONSULTAS	
Consulta 1	*Almendras*
Consulta 2	*Pistachos*
Consulta 3	*Tractores*
Consulta 4	*Aceite de oliva*
Consulta 5	*Ayuda ganaderos en Galicia*
Consulta 6	*Cereales*
Consulta 7	*Maíz*

Consulta 8	*Andalucía Doñana PP*
Consulta 9	*Luis Planas reparte ayudas*

Tabla 1: Consultas realizadas.

Con el propósito de evaluar de manera precisa y rigurosa los resultados obtenidos mediante las consultas realizadas, se ha tomado la decisión de utilizar un conjunto de medidas reconocidas en el ámbito de la recuperación de información. Estas medidas incluyen la Precisión, Recall, F1-Score, K-precisión y el MAP (Mean Average Precision).

La Precisión se define como la proporción de documentos relevantes recuperados sobre el total de documentos recuperados. Es una medida que indica la exactitud de los resultados obtenidos. El Recall, por su parte, representa la proporción de documentos relevantes recuperados sobre el total de documentos relevantes existentes. Mide la capacidad de recuperación de información relevante.

El F1-Score es una medida que combina la Precisión y el Recall en un solo valor. Proporciona una visión general del equilibrio entre ambas medidas, siendo útil para comparar diferentes sistemas o enfoques. Por otro lado, la K-precisión evalúa la relevancia de los primeros K documentos recuperados, lo cual es importante cuando se desea analizar los resultados más relevantes.

Finalmente, el MAP, o Mean Average Precision, calcula el promedio de las precisiones obtenidas en diferentes puntos de corte en los resultados de búsqueda. Esta medida proporciona una evaluación global del rendimiento del sistema de recuperación de información.

A continuación, se exponen las tablas que presentan los resultados obtenidos para cada consulta realizada. Es importante tener en cuenta que nuestro sistema devuelve por consulta un total de 25 documentos, y en la interfaz web se mostrarán únicamente los primeros 10 documentos, con opción de que se muestren más si lo requiere el usuario, por ello para las medidas K-precisión y MAP el valor de k tiene valor de 10:

Consulta 1: *Almendras*					
Doc. relevantes: 8					
-	Precisión	Recall	F1-Score	K-precision, k = 10	MAP, k=10
Modelo 1	*7/25*	*7/8*	*0.42*	*6/10*	*0.85*
Modelo 2	*8/25*	*8/8*	*0.48*	*5/10*	*0.88*
Modelo 3	*6/25*	*6/8*	*0.36*	*6/10*	*0.77*
Modelo 4	*6/25*	*6/8*	*0.36*	*5/10*	*0.52*

Modelo 5 con API Deepl	*8/25*	*8/8*	*0.48*	*8/10*	*0.98*
Modelo 5 con biblioteca de Python	*7/25*	*7/8*	*0.42*	*5/10*	*0.86*

Tabla 2: Resultados de la consulta 1.

Consulta 2: *Pistachos*					
Doc. relevantes: 2					
-	Precisión	Recall	F1-Score	K-precision, k = 10	MAP, k=10
Modelo 1	1/25	1/2	0.07	1/10	1
Modelo 2	2/25	2/2	0.14	1/10	1
Modelo 3	1/25	1/2	0.07	1/10	0.5
Modelo 4	1/25	1/2	0.07	1/10	1
Modelo 5 con API Deepl	2/25	2/2	0.14	1/10	1
Modelo 5 con biblioteca de Python	2/25	2/2	0.14	2/10	1

Tabla 3: Resultados de la consulta 2.

Consulta 3: *Tractores*					
Doc. relevantes: 12					
-	Precisión	Recall	F1-Score	K-precision, k = 10	MAP, k=10
Modelo 1	10/25	10/12	0.07	7/10	0.87
Modelo 2	8/25	8/12	0.43	5/10	0.72
Modelo 3	6/25	6/12	0.32	6/10	0.85
Modelo 4	10/25	10/12	0.07	6/10	0.86
Modelo 5 con API Deepl	9/25	9/12	0.48	7/10	0.93
Modelo 5 con biblioteca de Python	7/25	7/12	0.37	6/10	0.94

Tabla 4: Resultados de la consulta 3.

Consulta 4: *Aceite de oliva*					
Doc. relevantes: más de 25					
-	Precisión	Recall	F1-Score	K-precision, k = 10	MAP, k=10
Modelo 1	25/25	25/25	1	10/10	1
Modelo 2	25/25	25/25	1	10/10	1
Modelo 3	23/25	23/25	0.92	10/10	1
Modelo 4	24/25	24/25	0.96	10/10	1
Modelo 5 con API Deepl	25/25	25/25	1	10/10	1
Modelo 5 con biblioteca de Python	25/25	25/25	1	10/10	1

Tabla 5: Resultados de la consulta 4.

Consulta 5: *Ayudas ganaderos en Galicia*					
Doc. relevantes: 5					
-	Precisión	Recall	F1-Score	K-precision, k = 10	MAP, k=10
Modelo 1	2/25	2/5	0.13	1/10	1
Modelo 2	3/25	3/5	0.2	2/10	0.83
Modelo 3	2/25	2/5	0.13	1/10	1
Modelo 4	3/25	3/5	0.2	1/10	0.5
Modelo 5 con API Deepl	4/25	4/5	0.26	2/10	1
Modelo 5 con biblioteca de Python	5/25	5/5	0.33	4/10	0.93

Tabla 6: Resultados de la consulta 5.

Consulta 6: *Cereales*					
Doc. relevantes: más de 25					
-	Precisión	Recall	F1-Score	K-precision, k = 10	MAP, k=10
Modelo 1	24/25	24/25	0.96	10/10	1

Modelo 2	25/25	25/25	1	10/10	1
Modelo 3	20/25	20/25	0.8	10/10	1
Modelo 4	22/25	22/25	0.88	9/10	0.97
Modelo 5 con API Deepl	25/25	25/25	1	10/10	1
Modelo 5 con biblioteca de Python	25/25	25/25	1	10/10	1

Tabla 7: Resultados de la consulta 6.

Consulta 7: *Maíz*					
Doc. relevantes: 10					
-	Precisión	Recall	F1-Score	K-precision, k = 10	MAP, k=10
Modelo 1	6/25	6/10	0.34	6/10	0.85
Modelo 2	5/25	5/10	0.28	5/10	0.78
Modelo 3	8/25	8/10	0.45	5/10	0.90
Modelo 4	5/25	5/10	0.28	5/10	0.88
Modelo 5 con API Deepl	7/25	7/10	0.4	6/10	0.97
Modelo 5 con biblioteca de Python	5/25	5/10	0.28	4/10	0.73

Tabla 8: Resultados de la consulta 7.

Consulta 8: *Andalucía Doñana PP*					
Doc. relevantes: 23					
-	Precisión	Recall	F1-Score	K-precision, k = 10	MAP, k=10
Modelo 1	21/25	21/23	0.87	8/10	1
Modelo 2	19/25	19/23	0.78	9/10	0.98
Modelo 3	20/25	20/23	0.82	10/10	1
Modelo 4	23/25	23/23	0.95	6/10	1
Modelo 5 con API	23/25	23/23	0.95	10/10	1

Deepl					
Modelo 5 con biblioteca de Python	20/25	20/23	0.82	10/10	1

Tabla 9: Resultados de la consulta 8.

Consulta 9: *Luis Planas reparte ayudas*					
Doc. relevantes: más de 25					
-	Precisión	Recall	F1-Score	K-precision, k = 10	MAP, k=10
Modelo 1	15/25	15/25	0.6	7/10	0.93
Modelo 2	21/25	21/25	0.84	9/10	1
Modelo 3	6/25	6/25	0.24	5/10	0.63
Modelo 4	13/25	13/25	0.52	8/10	0.97
Modelo 5 con API Deepl	20/25	20/25	0.8	9/10	0.96
Modelo 5 con biblioteca de Python	17/25	17/25	0.68	8/10	0.91

Tabla 10: Resultados de la consulta 9.

Con el fin de obtener una comprensión más precisa y detallada del modelo que ha arrojado los mejores resultados en general, se sugiere dirigir la atención hacia la tabla 11. En dicha tabla, se presentan las medias totales para las métricas más importantes, permitiendo así una evaluación más fácil de visualizar.

Resultados generales			
-	Precisión	Recall	K-precision, k = 10
Modelo 1	0.49	0.74	0.62
Modelo 2	0.51	0.82	0.62
Modelo 3	0.32	0.64	0.6
Modelo 4	0.47	0.78	0.56
Modelo 5 con API Deepl	0.55	0.89	0.7
Modelo 5 con biblioteca de Python	0.5	0.83	0.65

Tabla 11: Resultados generales calculados a partir de las consultas.

Como se observa en la tabla 11, se evidencia de manera contundente que los resultados más destacados se obtienen al emplear un modelo de embeddings altamente especializado en el ámbito de la agricultura. Es importante destacar que dichos resultados superan a los demás en términos de rendimiento, al mismo tiempo que se logra reducir significativamente el tamaño en memoria requerido para su implementación. Además, al hacer uso de una API de traducción de alta competencia, se puede constatar que los resultados obtenidos son considerablemente superiores en comparación con el empleo de las traducciones proporcionadas por la biblioteca de Python. Esta observación pone de manifiesto la relevancia de utilizar herramientas especializadas y de calidad para lograr una precisión y fluidez óptimas en los procesos de traducción.

Resultados para elegir el score

Conforme se ha explicado previamente en la sección 3.5, al realizar una consulta a ElasticSearch, se obtiene un conjunto de noticias, cada una de las cuales se encuentra asociada a un score, el cual representa un valor numérico. Este score indica en qué medida el documento se ajusta a la consulta realizada, siendo más alto cuanto mayor sea la coincidencia. En vista de esto, se tomó la decisión de establecer un límite mínimo de score para considerar un documento como relevante. Por lo tanto, aquellos documentos cuyo score fuera inferior a dicho límite serían considerados no relevantes.

Aunque en algunos modelos o consultas pudiesen surgir casos en los que documentos con un score más elevado fueran considerados no relevantes en comparación con documentos de menor puntaje que se consideraron relevantes, se hizo necesario establecer un límite general para descartar aquellos documentos que, sin duda alguna, carecían de relevancia. Para tal fin, se realizaron varias consultas similares a las anteriores con el modelo de embeddings definitivo.

A continuación se presentan algunas de las consultas que resultaron ser de vital importancia para determinar el puntaje relevante:

Consulta: Cereales			
Posición en la consulta	***Doc. relevante o no***	***Título de la noticia***	***Score***
1	Doc. relevante	“Una lonja de cereales a golpe de clic y en el móvil”	1.85
2	Doc. relevante	“El Port de Tarragona registra un triple récord en movimientos de cereales en el primer trimestre”	1.80
X	Docs. relevantes	. . .	
27	Doc. no relevante	“Por un lenguaje inclusivo agrario”	1.56

Tabla 12: Resultados de la consulta ‘Cereales’ para obtener el parámetro score.

Consulta: Pistachos			
Posición en la consulta	***Doc. relevante o no***	***Título de la noticia***	***Score***
1	Doc. relevante	"Castilla la mancha aprobará un plan integral del pistacho y promoverá una IGP de este producto"	1.72
2	Doc. relevante	"La Universidad de Córdoba participa en un programa de mejor genética en pistachos"	1.70
3	Doc. no relevante	"España y otros 12 ven trato diferenciado en el veto al grano ucrariano"	1.68

Tabla 13: Resultados de la consulta 'Pistachos' para obtener el parámetro score.

Consulta: Maíz			
Posición en la consulta	***Doc. relevante o no***	***Título de la noticia***	***Score***
1	Doc. relevante	"Revive los mejores momentos del III Congreso del Maíz"	1.74
2	Doc. relevante	"Ciencia y agua para el futuro del maíz"	1.68
X	Docs. relevantes	. . .	
7	Doc. no relevante	"La temporada de caza en Extremadura se inicia este sábado con el rececho del corzo"	1.60

Tabla 14: Resultados de la consulta 'Maíz' para obtener el parámetro score.

Tras analizar detenidamente los resultados más significativos obtenidos por nuestro sistema, tal como se evidencia en las tablas anteriores, se ha constatado la complejidad que conlleva determinar el valor óptimo del score mínimo. Esta tarea se ve desafiada por el hecho de que la elección del score influye tanto en el rendimiento positivo como en negativo de diferentes consultas. Con el objetivo de abordar esta cuestión, se llevaron a cabo diversas pruebas para evaluar el desempeño del sistema utilizando diferentes score mínimos. Tras exhaustivas evaluaciones, se determinó que el valor de puntaje óptimo fue de 1.68, debido a los resultados superiores que demostró en comparación con otros valores.

ANEXO III. PROBLEMAS SIN SOLUCIÓN

En la presente sección se abordarán los desafíos que han surgido durante el desarrollo del sistema, así como las medidas adoptadas para mitigarlos. Se proporcionará un análisis detallado de los obstáculos encontrados en el proceso y se expondrán las estrategias implementadas con el propósito de contrarrestarlos de manera efectiva.

Etiquetas desbalancean los resultados

Conforme se expuso en la sección 3.1, inicialmente se planteó la hipótesis de que al seleccionar etiquetas para las noticias se lograría mejorar los resultados del motor de búsqueda. Sin embargo, esta suposición resultó no ser precisa debido a la naturaleza excesivamente generalista de dichas etiquetas (por ejemplo, "Agricultura", "Campo", entre otras). Esta situación condujo a una homogeneización de los embeddings generados en las consultas, lo cual a su vez desembocó en resultados de inferior calidad. Como consecuencia, se tomó la decisión de mantener la extracción de las etiquetas con la perspectiva de utilizarlas en un futuro para desarrollar un modelo de embeddings propio. Se espera que este enfoque proporcione información adicional que pueda ser aprovechada para entrenar dicho modelo de manera más efectiva y, de este modo, mejorar los resultados obtenidos en el proceso de búsqueda.

Descripciones cuando el contenido había cambiado

Como se ha expuesto en la sección 3.4, el módulo de revisión de noticias desempeña la función de verificar que el contenido de las noticias no haya sido actualizado. Este proceso se lleva a cabo de manera sencilla mediante la comparación de la fecha actual de la noticia con la fecha almacenada en la base de datos. A continuación, se procede a actualizar los atributos de la noticia de manera sistemática. No obstante, existe un atributo que no sigue esta dinámica, el cual corresponde a la descripción. La descripción consiste en un breve texto extraído de la página principal del portal de noticias, no de la propia noticia. El acceso a esta información se torna imposible debido a que, a medida que se agregan nuevas noticias al portal, tanto la noticia como su descripción pierden relevancia y se desplazan a páginas secundarias. Es por esta razón que se ha decidido que el contenido de la descripción no se modifique, independientemente de si la información de la noticia ha cambiado. Se asume que el contenido de la noticia no variará de manera significativa alterando drásticamente el valor de los embeddings utilizados.