

# Reporte sobre Revisión Sistemática en Optimización de Portafolios utilizando Inteligencia Artificial

## Seminario de Investigación I

Javier Horacio Pérez Ricárdez

20 de octubre del 2024

## 1 Introducción

En el presente informe, se detalla la revisión sistemática realizada sobre el tema de optimización de portafolios en el contexto de la inteligencia artificial. La búsqueda se enfocó en artículos publicados entre los años 2010 y 2024, con el objetivo de evaluar la cantidad y calidad de la literatura disponible en este ámbito.

## 2 Metodología

La búsqueda se llevó a cabo en diversas bases de datos académicas, a saber: Scopus, ScienceDirect, Wiley, MDPI y IEEE. Se utilizaron las palabras clave "*optimization*" y "*portfolio*" en los resúmenes de los artículos. A continuación, se presenta el número de registros obtenidos en cada una de estas fuentes:

- **Scopus:** 434 registros
- **ScienceDirect:** 5,618 registros
- **Wiley:** 182 registros
- **MDPI:** 109 registros
- **IEEE:** 192 registros

### 2.1 Total de Registros

En total, se obtuvieron **6,535 registros** a partir de las bases de datos mencionadas. Sin embargo, al revisar estos registros, se identificaron **5,801 artículos** que carecían de título, resumen o fecha. Esto dejó un total de **734 registros** que cumplían con los criterios básicos de inclusión.

## 2.2 Filtrado de Registros

Durante el proceso de revisión, se encontraron **4 registros duplicados**, los cuales fueron eliminados, resultando en **732 artículos** con título, resumen y fecha.

Adicionalmente, se evaluó la coherencia entre los títulos y resúmenes de los artículos. Se identificaron **189 registros** con baja coherencia y **543 registros** con alta coherencia. De estos, **383 artículos** cumplieron con los criterios de inclusión para la revisión sistemática.

## 2.3 Criterios de Inclusión

Los criterios de inclusión utilizados para seleccionar los artículos en esta revisión sistemática fueron los siguientes:

- **Relevancia Temática:** Los artículos debían abordar directamente la optimización de portafolios en el contexto de la inteligencia artificial.
- **Publicación entre 2010 y 2024:** Solo se consideraron artículos publicados dentro de este rango de fechas.
- **Disponibilidad de Título, Resumen y Fecha:** Se incluyeron solo aquellos artículos que contaban con un título, resumen y fecha de publicación completos.
- **Idioma:** Se aceptaron únicamente artículos escritos en inglés o español.
- **Calidad Metodológica:** Se priorizaron artículos que presentaban metodologías claras y rigurosas en su investigación.

## 3 Análisis de Registros por Año

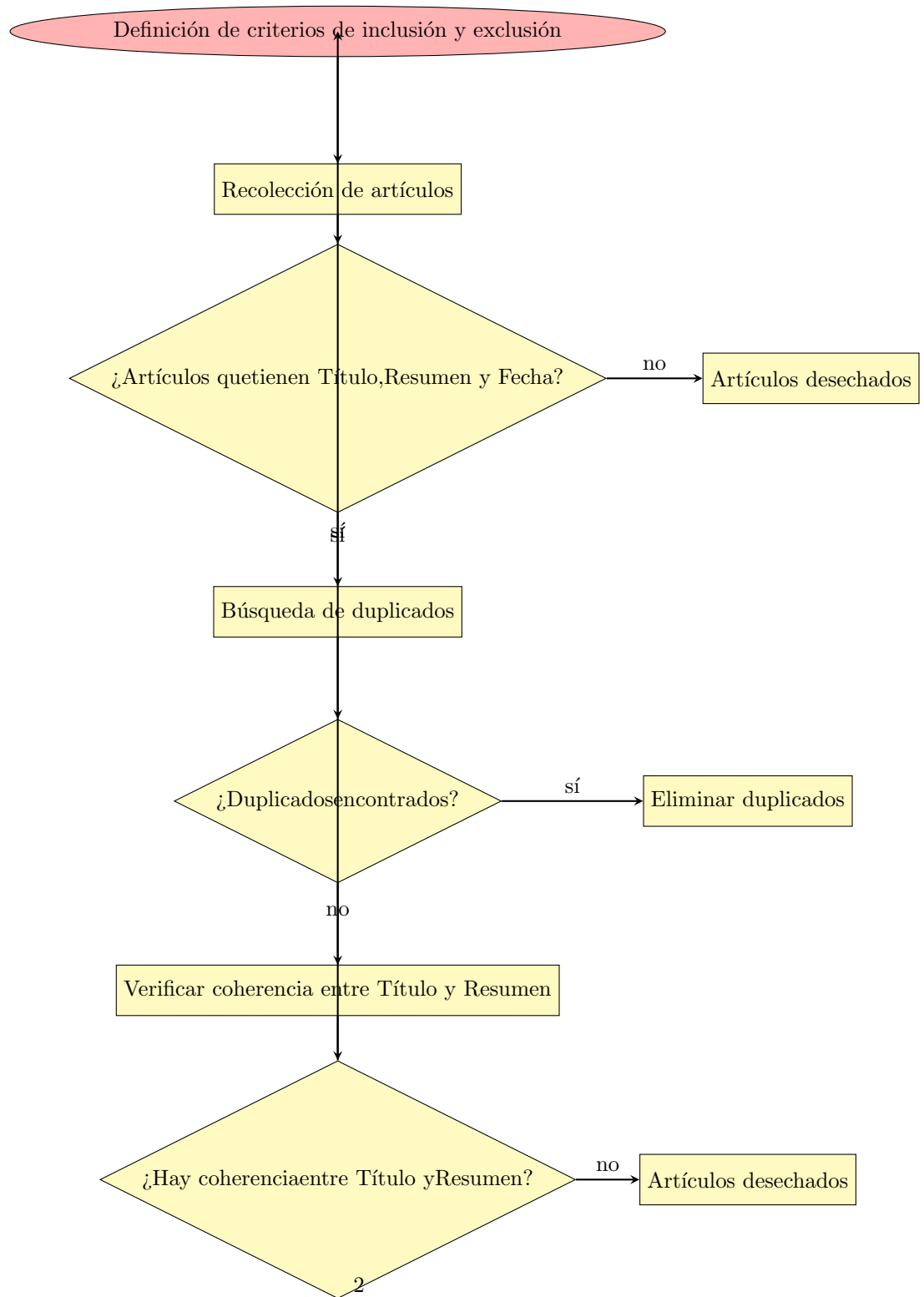
A continuación, se presenta el desglose de registros por año:

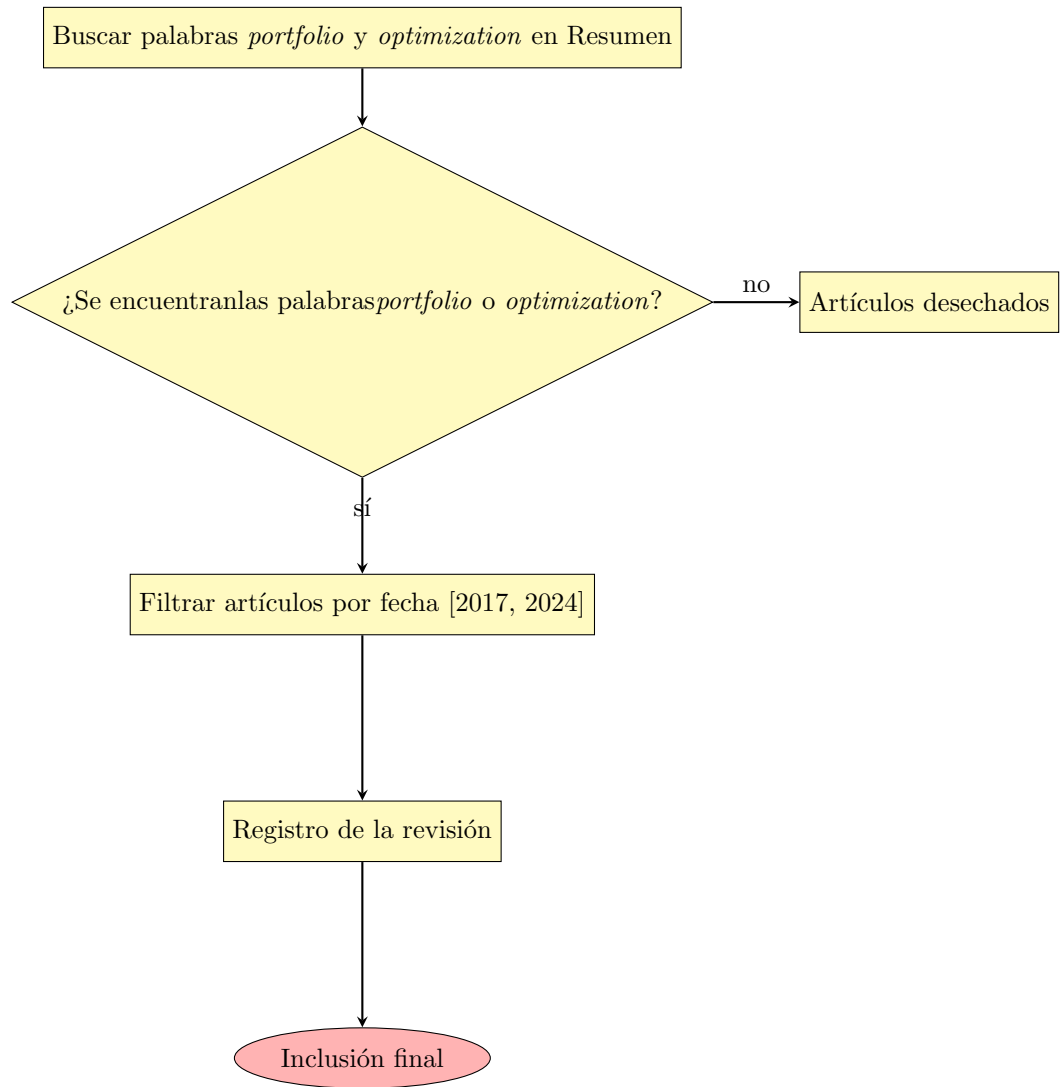
- **2017:** 25 registros
- **2018:** 26 registros
- **2019:** 35 registros
- **2020:** 34 registros
- **2021:** 64 registros
- **2022:** 51 registros
- **2023:** 78 registros
- **2024:** 70 registros

Este análisis temporal indica un creciente interés en la optimización de portafolios, especialmente en los años más recientes, lo que sugiere un desarrollo significativo en la investigación y aplicación de técnicas de inteligencia artificial en este campo.

## 4 Conclusiones

La revisión sistemática realizada ha permitido identificar un total de **383 artículos** relevantes sobre la optimización de portafolios en el contexto de la inteligencia artificial. Este cuerpo de literatura refleja un crecimiento constante en la investigación desde 2017 hasta 2024, lo que destaca la importancia y el potencial de la inteligencia artificial en la mejora de estrategias de inversión y gestión de portafolios.





# Descripción detallada del código en Streamlit para análisis de archivos .ris y .bib

## 1 Importaciones

El código comienza importando las siguientes bibliotecas:

- **streamlit**: Para construir la interfaz de usuario y mostrar los resultados en la aplicación web.
- **rispy**: Biblioteca para leer y procesar archivos **.ris**.
- **bibtexparser**: Para leer y procesar archivos **.bib**.
- **pandas**: Para manejar y analizar datos en forma de tablas (DataFrames).
- **TfidfVectorizer** y **cosine\_similarity**: Herramientas de **sklearn** para analizar y calcular la similitud entre textos (títulos y resúmenes).

## 2 Funciones

A continuación se describen las funciones implementadas en el código:

### 2.1 Función `filter_records(records)`

Esta función filtra los registros que contienen los campos válidos: **Título**, **Resumen** y **Fecha**. Los registros válidos se guardan en un **DataFrame** con las columnas correspondientes. Los registros que no contienen estos campos se almacenan en una lista separada para identificar errores o información faltante.

### 2.2 Función `parse_ris(files)`

Esta función toma los archivos **.ris** cargados y utiliza **rispy** para procesarlos. Luego, filtra los registros utilizando la función `filter_records()` y separa los válidos de los inválidos. Devuelve un **DataFrame** con los registros válidos y el número de registros inválidos.

### 2.3 Función `parse_bib(files)`

Esta función permite leer archivos `.bib`, extraer títulos, resúmenes y fechas, y filtrar registros de manera similar a los archivos `.ris`.

### 2.4 Función `find_duplicates(df)`

Esta función busca registros duplicados en el `DataFrame` en base a los campos `Título` y `Resumen`. Devuelve un `DataFrame` con los registros duplicados.

### 2.5 Función `check_similarity(df, threshold=0.2)`

La función calcula la similitud entre el título y el resumen de cada registro utilizando TF-IDF y la métrica de *similitud del coseno*, como se describe en el código.

### 2.6 Función `apply_selection_criteria(df)`

Esta función filtra los registros que contienen las palabras clave "portfolio" u "optimization" en el resumen. Luego, aplica un filtro adicional por fecha, seleccionando solo registros entre los años 2017 y 2024.

## 3 Interfaz en Streamlit

La interfaz de usuario se construye con `Streamlit`. El flujo general es el siguiente:

1. `st.file_uploader()` se utiliza para cargar archivos `.ris` y `.bib`.
2. Si se cargan archivos, estos se procesan con las funciones `parse_ris()` y `parse_bib()`.
3. Se busca duplicados utilizando `find_duplicates()`.
4. Se calcula la coherencia entre el título y el resumen usando `check_similarity()`.
5. Se aplican criterios de selección de palabras clave y fechas mediante la función `apply_selection_criteria()`.
6. Finalmente, se permite descargar los resultados filtrados en un archivo CSV.

## 4 Flujo General de la Aplicación

El flujo completo de la aplicación es el siguiente:

- **Cargar archivos .ris y .bib → Filtrar registros válidos e inválidos → Buscar duplicados → Verificar coherencia entre Título y Resumen → Aplicar criterios de selección → Descargar CSVs.**

Esta aplicación es útil para procesar referencias bibliográficas tanto en formato .ris como .bib, permitiendo una gestión más eficiente de registros.