Escuela Técnica Superior de
Ingeniería Informática

Trabajo Fin de Grado

Grado en Ingeniería Informática - Ingeniería del Software

Departamento de Ciencias de la Computación e Inteligencia Artificial

Mejorando el algoritmo Classifier Chains para clasificación multi-etiqueta mediante clustering difuso

Autor/a:
D. Luis Mellado Díaz

Tutor:
Dr. Jose María Moyano Murillo

Sevilla, 28 de Mayo de 2025

Índice general

Índice de figuras

Índice de tablas

1. Introducción

El *aprendizaje multi-etiqueta* (*multi-label classification*) aborda el desafío de clasificar instancias que pueden pertenecer simultáneamente a múltiples categorías o etiquetas, en contraste con los enfoques tradicionales que asignan una única etiqueta a cada instancia [1]. En la clasificación tradicional, cada instancia se asigna a una sola etiqueta, como se ilustra en la Figura 1.1. Sin embargo, en la clasificación multi-etiqueta, una misma instancia puede pertenecer a varias categorías simultáneamente, lo que complica el proceso de aprendizaje y evaluación (Figura 1.2).

Figura 1.1: Ejemplo de clasificación tradicional.

Este tipo de clasificación presenta características únicas que han generado el desarrollo de métricas personalizadas, técnicas de preprocesamiento específicas y algoritmos especializados. Las peculiaridades de los conjuntos de datos *multi-etiqueta* incluyen alta dimensionalidad, desequilibrios en la frecuencia de las etiquetas y relaciones complejas entre estas, lo que plantea retos adicionales [1].

El modelo **Classifier Chains (CC)** es una técnica ampliamente utilizada en la clasificación *multi-etiqueta* debido a su capacidad para capturar dependencias entre etiquetas. A diferencia del método de **Binary Relevance (BR)**, que trata cada etiqueta de forma independiente [2], **CC** organiza las etiquetas en una cadena, donde

Figura 1.2: Ejemplo de clasificación multi-etiqueta.

cada clasificador binario predice la presencia o ausencia de una etiqueta utilizando tanto las características de entrada como las predicciones de las etiquetas anteriores en la cadena. Este enfoque permite incorporar información contextual relevante, lo que mejora la capacidad del modelo para predecir combinaciones coherentes de etiquetas [3].

Sin embargo, **CC** enfrenta limitaciones importantes, especialmente en problemas donde el orden de las etiquetas en la cadena tiene un impacto significativo en el rendimiento predictivo [4]. Además, no aborda explícitamente la relación estructural entre conjuntos de etiquetas que podrían agruparse de manera natural o semántica. Estas deficiencias motivan la exploración de enfoques que superen estas limitaciones, mejorando la precisión predictiva y la robustez del modelo. En este contexto, el enfoque **Label Clusters Chains Multi-Label Classifier (LCC-MLC)** [5] propone una mejora al CC tradicional al agrupar etiquetas correlacionadas en clústeres y construir cadenas de clasificadores basadas en estos grupos. Este método permite capturar relaciones más complejas entre etiquetas, pero aún presenta limitaciones, como la asignación rígida de etiquetas a un único clúster.

Para abordar estas limitaciones, este trabajo propone la integración de técnicas de clustering difuso en el modelo **LCC-MLC**. El clustering difuso permite que una etiqueta pertenezca a múltiples clústeres con diferentes grados de pertenencia, lo que refleja de manera más precisa la incertidumbre y las relaciones parciales entre etiquetas. Este enfoque, denominado **Fuzzy Cluster-Based Classifier Chain (FCCC)**, busca mejorar la capacidad del modelo para capturar dependencias complejas entre etiquetas y, en última instancia, aumentar la precisión predictiva en problemas de clasificación multi-etiqueta.

El resto de la memoria se organiza del siguiente modo:

2. Estado del arte. Se presentan los conceptos fundamentales sobre clasificación

multi-etiqueta y *clustering difuso*. Se describen los enfoques tradicionales para la clasificación *multi-etiqueta*, los métodos de transformación de problemas y los modelos existentes en la literatura, incluyendo **LCC-MLC** [5], que sirve de base para la propuesta de este trabajo.

3. Propuesta. Se describe el modelo propuesto **FCCC**, que incorpora técnicas de *clustering difuso* para mejorar la clasificación *multi-etiqueta*. Se detallan las fases del algoritmo, incluyendo la generación de *clústeres difusos*, la secuenciación de *clústeres*, el entrenamiento de clasificadores y el esquema de predicción final.

4. Estudio Experimental. Se presentan los experimentos realizados para evaluar el desempeño del modelo propuesto en comparación con otros enfoques existentes. Se describen la configuración experimental, los conjuntos de datos utilizados y los resultados obtenidos.

5. Conclusiones y Trabajo Futuro. Se exponen las conclusiones obtenidas a partir del estudio y se proponen futuras líneas de investigación para mejorar y extender el modelo.

2. Estado del Arte

En esta sección, se presentará un recorrido por los conceptos clave y enfoques más relevantes relacionados con el *aprendizaje multietiqueta* y el *clustering difuso*. Comenzaremos con una descripción general del *aprendizaje multi-etiqueta*, explicando los fundamentos de esta metodología y los principales métodos utilizados para abordarla. Posteriormente, se trata el modelo **Label Clusters Chains Multi-Label Classifier** (**LCC-MLC**), que sirve como base para la propuesta de este trabajo. Finalmente, definiremos el concepto y utilidad de un *conjunto difuso*.

2.1. Aprendizaje Multi-etiqueta

El *aprendizaje multi-etiqueta* es un problema de *aprendizaje supervisado* en el que cada instancia puede estar asociada con múltiples etiquetas simultáneamente. A diferencia de la *clasificación de una sola etiqueta*, donde una instancia está vinculada a una única clase o etiqueta, en el *aprendizaje multi-etiqueta* una instancia está asociada a un subconjunto de etiquetas de un conjunto total $\mathcal{L} = \{\lambda_1, \lambda_2, \ldots, \lambda_q\}$. Esto puede representarse mediante un vector binario $\mathbf{y} \in \{0,1\}^q$, donde $y_j = 1$ indica que la etiqueta λ_j está presente en la instancia y $y_j = 0$ lo contrario [6].

Formalmente, dado un conjunto de datos de entrenamiento $D = \{(x_i, \mathbf{y}_i) \mid i = 1, 2, \ldots, N\}$, donde $x_i \in \mathbb{R}^d$ representa las características de una instancia e $\mathbf{y}_i \in \{0,1\}^q$ denota el conjunto de etiquetas asociadas, el objetivo es aprender una función $h : \mathbb{R}^d \to \{0,1\}^q$ que pueda predecir el conjunto de etiquetas $\mathbf{y}$ para nuevas instancias [6].

Los enfoques de *aprendizaje multi-etiqueta* se dividen generalmente en métodos de adaptación de algoritmos y **métodos de transformación de problemas**. A continuación nos centraremos en este último enfoque.

2.2. Métodos de Transformación de Problemas

En el ámbito del aprendizaje multi-etiqueta, los métodos de transformación de problemas son enfoques que convierten el problema original en uno o varios problemas de clasificación más simples, que pueden ser resueltos utilizando técnicas estándar de aprendizaje supervisado. A continuación, se describen algunos de los métodos más utilizados en la literatura, como **Binary Relevance** (**BR**), **Label Powerset** (**LP**) y **Classifier Chains** (**CC**). Para una revisión más exhaustiva de estos y otros métodos, se puede consultar [7], donde se presenta un análisis detallado de las técnicas de transformación de problemas en clasificación multietiqueta.

Binary Relevance (BR)

El método **BR** transforma el conjunto de datos en q conjuntos de datos y entrena q clasificadores binarios $H : X \to \{l, \neg l\}$ para cada $l \in \mathcal{L}$.

Durante este proceso, la información sobre las dependencias entre etiquetas no se conserva, lo que puede afectar la calidad de la predicción [2]. En resumen, se entrena un clasificador binario independiente para cada etiqueta. Aunque es sencillo y escalable, no modela dependencias entre etiquetas [6].

Label Powerset (LP)

LP transforma el problema en una *clasificación de una sola etiqueta*, donde cada combinación única de etiquetas se trata como una nueva clase. Después de la transformación, se entrena un clasificador de etiqueta única $H : X \to \mathcal{P}(\mathcal{L})$, donde $\mathcal{P}(\mathcal{L})$ es el conjunto potencia de todas las etiquetas en $\mathcal{L}$ [6].

El principal inconveniente de este enfoque es que el número de combinaciones de etiquetas y que el costo computacional crece exponencialmente con el número de etiquetas.

Classifier Chains (CC)

CC entrena q clasificadores, de manera similar a **BR**, pero enlazados en una cadena a través del espacio de características [3]. La clasificación comienza con el primer clasificador C_1 y avanza hasta el último C_q, pasando la información sobre las etiquetas entre los clasificadores. Este enfoque permite capturar dependencias entre etiquetas, ya que cada clasificador utiliza las predicciones de los clasifica-

dores anteriores en la cadena como características adicionales. Sin embargo, una limitación importante de **CC** es que el orden de las etiquetas en la cadena puede afectar significativamente el rendimiento del modelo, ya que las dependencias entre etiquetas no siempre siguen un orden predefinido.

Para abordar esta limitación, se propuso el uso de **Ensembles of Classifier Chains** (**ECC**) [4]. **ECC** consiste en entrenar un conjunto de m clasificadores, cada uno con una cadena de etiquetas ordenada aleatoriamente. Al combinar las predicciones de estos clasificadores, se reduce la dependencia del orden de las etiquetas y se mejora la robustez del modelo. Este enfoque permite capturar una mayor variedad de dependencias entre etiquetas, lo que puede resultar en una mejora significativa en la precisión predictiva, especialmente en conjuntos de datos donde las relaciones entre etiquetas son complejas y no lineales.

2.3. Label Clusters Chains Multi-Label Classifier (LCC-MLC)

El método **LCC-MLC** es una extensión del modelo de **CC** que busca mejorar la clasificación multi-etiqueta al aprovechar las correlaciones entre etiquetas. En lugar de procesar cada etiqueta de manera independiente o en una cadena arbitraria, **LCC-MLC** organiza las etiquetas en *clústeres* basados en sus relaciones semánticas y estadísticas.

El objetivo de **LCC-MLC** es mejorar la capacidad predictiva del modelo al reducir la propagación de errores característica de **CC** y capturar mejor las dependencias entre etiquetas. Para ello, sigue un proceso estructurado que incluye la detección de correlaciones entre etiquetas, la formación de clústeres disjuntos y la ordenación de estos en una cadena de clasificación.

El algoritmo **LCC-MLC** se compone de varias fases clave:

- **Paso 1: Modelo de Correlación de Etiquetas**
  Se calcula una medida de similitud entre etiquetas, como el *coeficiente de Jaccard*, para determinar qué etiquetas presentan una mayor relación entre sí.
- **Paso 2: Generación de Clústeres**

  1. Se construye una matriz de similitud $M = q \times q$ basada en las correlaciones entre etiquetas.
  2. Se transforma la matriz de similitud en una matriz de disimilitud para su uso en un algoritmo de agrupamiento jerárquico aglomerativo (**AHCA**) [8].

3. Se genera un dendrograma que representa la jerarquía de los *clústeres*.

- **Paso 3: Selección de la Mejor Partición**
  Para determinar la mejor agrupación de etiquetas, se emplea el *coeficiente de silueta* [9], que mide la calidad del agrupamiento.

- **Paso 4: Entrenamiento del Modelo**

  - **Secuencia de Clústeres:** Se ordenan los *clústeres* siguiendo la estructura del dendrograma generado.
  - **Uso de Etiquetas como Características:** Las etiquetas predichas en los *clústeres* anteriores se incorporan como nuevas características para mejorar la clasificación de los siguientes *clústeres*.
  - **Modelos Individuales por Clúster:** Se entrena un clasificador basado en **Random Forest** para cada *clúster*, permitiendo que las etiquetas más relacionadas se procesen juntas.

- **Paso 5: Salida del Entrenamiento**
  Cada *clúster* genera un conjunto de etiquetas predichas, las cuales se combinan para obtener la predicción multi-etiqueta final [5].

  A pesar de sus ventajas, el modelo **LCC-MLC** presenta algunas limitaciones importantes que motivan la inclusión del **clustering difuso** en nuestro enfoque:

  - **Agrupación rígida: LCC-MLC** asigna cada etiqueta a un único clúster, lo que no refleja adecuadamente la realidad en la que muchas etiquetas pueden estar relacionadas con múltiples grupos.
  - **Dependencia del orden de los clústeres:** El rendimiento del modelo puede verse afectado por el orden en el que se procesan los *clústeres*, ya que las etiquetas predichas en los primeros *clústeres* afectan a las posteriores.

  Para abordar estas limitaciones, proponemos la integración del **clustering difuso** en el modelo, permitiendo que una misma etiqueta pertenezca a múltiples *clústeres* con distintos grados de pertenencia. Este enfoque mejora la flexibilidad del modelo y captura mejor la incertidumbre en la clasificación multi-etiqueta, optimizando la capacidad predictiva de la cadena de clasificadores.

2.4. Clustering Difuso

El *clustering difuso* es una clase de algoritmos de agrupamiento en la que cada elemento puede pertenecer a múltiples grupos (*clusters*) con distintos

grados de pertenencia. Este enfoque surge para superar una limitación del agrupamiento tradicional (*hard clustering*), donde cada elemento es asignado de manera exclusiva a un solo clúster, sin considerar posibles similitudes con otros grupos.

Sea X un conjunto de datos, donde cada elemento $x \in X$ representa un objeto en el espacio de análisis. En este contexto, un *cluster difuso* A en X se define mediante una función de pertenencia $f_A : X \to [0, 1]$, la cual asigna a cada punto $x \in X$ un valor en el intervalo $[0, 1]$. Este valor, denotado como $f_A(x)$, representa el *grado de pertenencia* de x al clúster A. Cuanto más cercano sea $f_A(x)$ a la unidad, mayor será la pertenencia de x a A [10].

Uno de los algoritmos más utilizados en el *clustering difuso* es el *Fuzzy C-Means* (**FCM**) [11]. Este método es una extensión del algoritmo *k-means*, pero en lugar de asignar cada punto de datos de manera exclusiva a un único clúster, permite que cada punto pertenezca a múltiples clústeres con diferentes grados de pertenencia.

El algoritmo **FCM** se basa en la minimización de la siguiente función objetivo:

$$J_m = \sum_{i=1}^{N} \sum_{j=1}^{C} u_{ij}^m \, d^2(x_i, c_j), \tag{2.1}$$

donde:

- N es el número total de puntos de datos,
- C es el número de clústeres,
- $u_{ij} \in [0, 1]$ representa el grado de pertenencia del punto x_i al clúster j,
- $m > 1$ es un parámetro de difusidad que controla la nitidez de la clasificación,
- c_j es el centro del clúster j,
- $d(x_i, c_j)$ es la distancia (generalmente euclidiana) entre el punto x_i y el centro del clúster c_j.

El algoritmo **FCM** sigue un procedimiento iterativo que actualiza los valores de pertenencia y los centroides de los clústeres hasta converger a una solución óptima.

1. Inicializar aleatoriamente los grados de pertenencia u_{ij}.
2. Calcular los centroides de los clústeres según:
3. Actualizar los valores de pertenencia u_{ij} utilizando:
4. Repetir los pasos 2 y 3 hasta que la variación en los valores de pertenencia sea menor que un umbral predefinido.

Este enfoque permite una mayor flexibilidad en la asignación de los datos, lo que lo hace especialmente útil en problemas donde los límites entre clústeres no son claramente definidos.

3. Fuzzy Cluster-Based Classifier Chain

En esta sección, se describe en detalle el algoritmo **Fuzzy Cluster-Based Classifier Chain (FCCC)**, **propuesto en este TFG**, que combina técnicas de clustering difuso con cadenas de clasificadores para mejorar la clasificación multi-etiqueta. Esta propuesta busca superar las limitaciones de métodos previos, como LCC-MLC, al permitir que las etiquetas pertenezcan a múltiples clústeres con diferentes grados de pertenencia, capturando mejor las relaciones complejas entre ellas.

Además de la implementación de FCCC, en este trabajo se ha desarrollado y analizado el modelo **LCC-MLC**, lo que aporta un mayor valor al estudio comparativo entre distintas estrategias de clasificación multi-etiqueta. Todo el código fuente de estos algoritmos, junto con los experimentos realizados, está disponible en el siguiente repositorio de GitHub: `https://github.com/LuisMelladoDiaz/Difuse-Cluster-Chain-for-Multi-Label-Classification`.

Cabe destacar que, si bien el código ha sido desarrollado íntegramente por el autor, se ha contado con el apoyo de herramientas de inteligencia artificial para optimizar el proceso de desarrollo. En concreto, se utilizó **ChatGPT**[1] como asistencia para la resolución de errores (bugs) y el entorno de desarrollo **Cursor IDE**[2] para facilitar tareas de refactorización y limpieza del código.

El algoritmo FCCC se estructura en varias etapas clave:

1. Generación de la matriz de similitud entre etiquetas.
2. Aplicación de Fuzzy C-Means para clustering difuso.
3. Limpieza y normalización de la matriz de pertenencia.
4. Ordenación de la cadena de clústeres.
5. Entrenamiento de clasificadores por clúster.
6. Predicción y combinación de resultados.

[1] https://chat.openai.com
[2] https://www.cursor.com/

3.1. Generación de la matriz de similitud entre etiquetas

El primer paso del algoritmo consiste en calcular la similitud entre las etiquetas del conjunto de datos. Para ello, se utiliza el **índice de Jaccard**. La similitud entre dos etiquetas λ_i y λ_j se define como:

$$\text{Similitud}(\lambda_i, \lambda_j) = 1 - \frac{|\lambda_i \cap \lambda_j|}{|\lambda_i \cup \lambda_j|}$$

donde $|\lambda_i \cap \lambda_j|$ es el número de instancias que tienen ambas etiquetas, y $|\lambda_i \cup \lambda_j|$ es el número de instancias que tienen al menos una de las dos etiquetas.

Una vez calculada la similitud entre cada pareja de etiquetas se construye la matriz de similitud (ver Tabla 3.1), que permite identificar qué etiquetas están más relacionadas entre sí. Esta matriz será utilizada posteriormente para agrupar las etiquetas en clústeres difusos.

	λ_1	λ_2	λ_3	$\cdots$	λ_n
λ_1	$s(\lambda_1, \lambda_1)$	$s(\lambda_1, \lambda_2)$	$s(\lambda_1, \lambda_3)$	$\cdots$	$s(\lambda_1, \lambda_n)$
λ_2	$s(\lambda_2, \lambda_1)$	$s(\lambda_2, \lambda_2)$	$s(\lambda_2, \lambda_3)$	$\cdots$	$s(\lambda_2, \lambda_n)$
λ_3	$s(\lambda_3, \lambda_1)$	$s(\lambda_3, \lambda_2)$	$s(\lambda_3, \lambda_3)$	$\cdots$	$s(\lambda_3, \lambda_n)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
λ_n	$s(\lambda_n, \lambda_1)$	$s(\lambda_n, \lambda_2)$	$s(\lambda_n, \lambda_3)$	$\cdots$	$s(\lambda_n, \lambda_n)$

Tabla 3.1: Matriz de similitud para un conjunto con n etiquetas.

3.2. Aplicación de Fuzzy C-Means para clustering difuso

Una vez obtenida la matriz de similitud, se aplica el algoritmo **Fuzzy C-Means (FCM)** para agrupar las etiquetas en clústeres difusos. El clustering difuso permite que una etiqueta pertenezca a varios clústeres con diferentes grados de pertenencia.

El resultado es una **matriz de pertenencia** que indica el grado de pertenencia de cada etiqueta a cada clúster. Esta matriz se define como:

$$U = [u_{ij}]$$

donde u_{ij} representa el grado de pertenencia de la etiqueta i al clúster j. Los valores de u_{ij} están en el intervalo $[0, 1]$, y la suma de los grados de pertenencia de una etiqueta a todos los clústeres es igual a 1.

En la Tabla 3.2 se muestra un ejemplo de la matriz de pertenencia transpuesta U^T, donde las filas representan las etiquetas y las columnas representan los clústeres:

	C1	C2	$\cdots$	Ck
λ_1	u_{11}	u_{12}	$\cdots$	u_{1k}
λ_2	u_{21}	u_{22}	$\cdots$	u_{2k}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
λ_n	u_{n1}	u_{n2}	$\cdots$	u_{nk}

Tabla 3.2: Ejemplo de la matriz de pertenencia transpuesta U^T. Cada valor u_{ij} representa el grado de pertenencia de la etiqueta λ_i al clúster C_j.

3.3. Limpieza y normalización de la matriz de pertenencia

Para evitar que etiquetas con grados de pertenencia muy bajos afecten negativamente al modelo, se aplica un umbral sobre la matriz de pertenencia transpuesta U^T. Primero, se eliminan aquellos valores de pertenencia que están por debajo de un umbral predefinido (por ejemplo, 0.1) y, posteriormente, se normalizan las filas de la matriz para que la suma de los grados de pertenencia de cada etiqueta siga siendo 1. Este paso es crucial por dos razones principales:

1. **Reducción de la complejidad del modelo:** Si todos los *clústeres* incluyeran todas las etiquetas, el número de cálculos y la dimensionalidad del modelo aumentarían significativamente, lo que afectaría tanto el tiempo de entrenamiento como la eficiencia del clasificador.
2. **Filtrado de etiquetas irrelevantes:** Las etiquetas con un grado de pertenencia muy bajo en un *clúster* apenas influirían en la predicción final. Su eliminación permite que el modelo se centre en aquellas etiquetas que realmente aportan información relevante en cada agrupación.

De este modo, la limpieza y normalización de la matriz de pertenencia contribuye a mejorar la eficiencia del algoritmo y a garantizar que solo las etiquetas con una relación significativa con un *clúster* sean consideradas en las siguientes etapas del proceso.

3.4. Ordenación de la cadena de clústeres

Es necesario establecer un orden para procesar los clústeres, ya que dicho orden puede influir significativamente en el rendimiento del modelo, especialmente en problemas donde las etiquetas presentan dependencias complejas. A continuación, se describen dos enfoques distintos para la ordenación de los clústeres.

3.4.1. Ordenación aleatoria

Formalmente, sea $\mathcal{C} = \{C_1, C_2, C_3, \ldots, C_n\}$ el conjunto de clústeres obtenidos. Una permutación aleatoria de $\mathcal{C}$ se define como una reordenación de sus elementos, denotada por:

$$\sigma(\mathcal{C}) = (C_{\sigma(1)}, C_{\sigma(2)}, C_{\sigma(3)}, \ldots, C_{\sigma(n)})$$

donde σ es una función de permutación que asigna a cada índice i un nuevo índice $\sigma(i)$.

3.4.2. Ordenación por grado de dependencia

Se propone un método de ordenación de los clústeres basado en el grado de dependencia entre ellos, utilizando tanto la matriz de pertenencia transpuesta $\mathbf{U}^\top \in \mathbb{R}^{n \times m}$, donde n es el número de clústeres y m el número de etiquetas, como la matriz de similitud entre etiquetas $\mathbf{S}^\lambda \in \mathbb{R}^{m \times m}$, previamente calculada mediante el índice de Jaccard.

Cada entrada u_{ij} en $\mathbf{U}^\top$ representa el grado de pertenencia de la etiqueta λ_j al clúster C_i, mientras que S^λ_{ab} mide la similitud entre las etiquetas λ_a y λ_b.

A partir de estas matrices, se define la **similitud entre clústeres** C_i y C_j como:

$$S_{ij} = \frac{\sum_{a=1}^{m} \sum_{b=1}^{m} u_{ia} \cdot u_{jb} \cdot S^\lambda_{ab}}{\sum_{a=1}^{m} \sum_{b=1}^{m} u_{ia} \cdot u_{jb}},$$

donde el numerador representa una suma ponderada de similitudes entre etiquetas en función de su pertenencia a los clústeres C_i y C_j, y el denominador actúa como factor de normalización. Si el denominador es cero (por ejemplo, si los clústeres no comparten etiquetas), se define $S_{ij} = 0$.

Con la matriz resultante $\mathbf{S} \in \mathbb{R}^{n \times n}$, se construye una secuencia ordenada de clústeres utilizando el siguiente procedimiento:

1. Se selecciona como primer clúster aquel con **menor similitud media** respecto al resto (es decir, el más independiente).
2. En cada paso posterior, se elige el siguiente clúster como aquel que presenta **mayor similitud con el último clúster añadido**, permitiendo un recorrido progresivo desde clústeres más independientes hacia clústeres más dependientes.

Este tipo de ordenación favorece una cadena de clasificación en la que los primeros modelos se centran en patrones más generales, mientras que los posteriores pueden afinar la predicción teniendo en cuenta la información de etiquetas ya procesadas.

3.5. Entrenamiento de clasificadores por clúster

Para cada clúster, se entrena un clasificador independiente utilizando las etiquetas que tienen un grado de pertenencia significativo al clúster.

Dado que se trabaja con etiquetas relacionadas entre sí, es importante emplear un método de clasificación capaz de capturar estas dependencias. Cada clasificador se entrena utilizando tanto las características originales como las predicciones de las etiquetas que aparecían en los clústeres anteriores, lo que permite modelar la relación entre etiquetas de manera más efectiva.

Como una misma etiqueta puede pertenecer a varios clústeres con distintos grados, su predicción final se calculará combinando las salidas de cada clasificador de forma ponderada. Esta combinación se formaliza mediante la Ecuación 3.1:

$$\hat{Y}_j = \frac{\sum_{k=1}^{K} u_{jk} \cdot \hat{Y}_{jk}}{\sum_{k=1}^{K} u_{jk}} \tag{3.1}$$

donde $\hat{Y}_{jk}$ es la predicción del clasificador entrenado sobre el clúster k para la etiqueta j, y u_{jk} representa el grado de pertenencia de la etiqueta al clúster.

3.6. Predicción y combinación de resultados

Finalmente los clasificadores de cada clúster se utilizan para realizar predicciones. A continuación, se muestra un ejemplo general para la etiqueta λ_1. En este caso, se asume que su pertenencia a tres clústeres se describe mediante los valores α, β y γ, como se indica en la Ecuación 3.2:

$$\lambda_1 \rightarrow \begin{cases} C_1 & u_{11} = \alpha \\ C_2 & u_{12} = \beta \\ C_3 & u_{13} = \gamma \end{cases} \tag{3.2}$$

Los clasificadores correspondientes producen predicciones probabilísticas p_1, p_2 y p_3 para la etiqueta λ_1, como se muestra en la Ecuación 3.3:

$$\text{Predicción}(\lambda_1) = \begin{cases} \text{Clasificador}_1 \rightarrow p_1 \\ \text{Clasificador}_2 \rightarrow p_2 \\ \text{Clasificador}_3 \rightarrow p_3 \end{cases} \tag{3.3}$$

A continuación, estas predicciones se combinan de forma ponderada según los valores de pertenencia, obteniendo así el valor p, tal como se calcula en la Ecuación 3.4:

$$p = p_1\alpha + p_2\beta + p_3\gamma \tag{3.4}$$

Finalmente, se aplica un umbral de decisión θ sobre el valor combinado p, como se describe en la Ecuación 3.5:

$$\text{Predicción final} = \begin{cases} 1 & \text{si } p \geq \theta \\ 0 & \text{en caso contrario} \end{cases} \tag{3.5}$$

De este modo, la etiqueta λ_1 será asignada a la instancia si la suma ponderada supera el umbral θ.

4. Estudio experimental

El objetivo de los estudios experimentales es comparar el modelo Fuzzy Cluster-Based Classifier Chain (FCCC) propuesto con otros algoritmos de vanguardia en clasificación multi-etiqueta, utilizando una variedad de conjuntos de datos y métricas de evaluación. En esta sección, se presentan en primer lugar los conjuntos de datos multi-etiqueta utilizados en los experimentos, junto con sus características principales. A continuación, se describen las métricas de evaluación empleadas para medir el rendimiento de los algoritmos. Finalmente, se detallan los ajustes y configuraciones experimentales utilizados para garantizar una comparación justa y reproducible entre los diferentes enfoques. Los resultados obtenidos permitirán evaluar la efectividad de la propuesta en comparación con los métodos existentes, destacando sus ventajas y posibles limitaciones.

4.1. Configuración de los experimentos

4.1.1. Conjuntos de datos

Los experimentos se llevaron a cabo utilizando una colección variada de 10 conjuntos de datos de referencia, abarcando múltiples áreas, como clasificación de texto, multimedia, química y biología. Estos conjuntos de datos están disponibles en la web del grupo de investigación KDIS: `http://www.uco.es/kdis/mllresources/`.

La Tabla 4.1 muestra estos conjuntos de datos junto con sus principales características, incluyendo el dominio, el número de instancias (m), el número de etiquetas (q), la cantidad de características (d), la cardinalidad (card, que representa el número medio de etiquetas por instancia) y la densidad (dens, obtenida al dividir la cardinalidad entre el número total de etiquetas). Los conjuntos de datos están ordenados de acuerdo con la cantidad de etiquetas.

Tabla 4.1: Resumen de las características de los conjuntos de datos

Conjunto de datos	Dominio	m	q	d	card	dens
Emotions	Música	593	6	72	1.868	0.311
CHD_49	Medicina	555	6	49	2.580	0.430
VirusGO	Biología	207	6	749	1.217	0.203
Foodtruck	Recomend.	407	12	21	2.290	0.191
Water-quality	Química	1060	14	16	5.073	0.362
Birds	Audio	645	19	260	1.014	0.053
Yahoo_Arts	Texto	7484	26	23150	1.654	0.064
Yahoo_Business	Texto	11210	30	21920	1.599	0.053
Mediamill	Video	43910	101	120	4.376	0.043
BibTex	Text	7395	159	1836	2.402	0.015

4.1.2. Métricas de evaluación

Para evaluar el rendimiento de los métodos de clasificación multi-etiqueta, es fundamental utilizar métricas que tengan en cuenta la naturaleza específica de este tipo de problemas. En este estudio, se han seleccionado varias métricas ampliamente utilizadas en la literatura, las cuales permiten medir diferentes aspectos del rendimiento de los modelos. Estas métricas se han implementado utilizando la biblioteca **scikit-learn** [12].

Subset Accuracy

Esta métrica, también conocida como **Exact Match**, mide la proporción de instancias en las que todas las etiquetas predichas coinciden exactamente con las etiquetas reales. Es una métrica estricta, ya que requiere que todas las etiquetas sean correctas para considerar la predicción como acertada. Se define como:

$$\uparrow \text{Subset Accuracy} = \frac{1}{N}\sum_{i=1}^{N}[\![Y_i = \hat{Y}_i]\!]$$

donde Y_i es el conjunto de etiquetas reales, $\hat{Y}_i$ es el conjunto de etiquetas predichas, y $[\![\pi]\!]$ es una función que devuelve 1 si el predicado π es verdadero, y 0 en caso contrario.

Hamming Loss

Esta métrica mide la fracción de etiquetas incorrectamente predichas respecto al total de etiquetas. A diferencia de la exactitud, es una métrica que se

minimiza, y es útil para evaluar el rendimiento a nivel de etiqueta individual. Se define como:

$$\downarrow \text{Hamming Loss} = \frac{1}{N \cdot q} \sum_{i=1}^{N} \sum_{j=1}^{q} [\![Y_{ij} \neq \hat{Y}_{ij}]\!]$$

donde q es el número total de etiquetas.

F1-Score (Macro)

El F1-score macro es la media armónica de la precisión y el recall macro, calculados para cada etiqueta individualmente. Esta métrica es especialmente útil cuando se desea dar igual importancia a todas las etiquetas, independientemente de su frecuencia.

$$\uparrow \text{F1-Score (Macro)} = \frac{1}{q} \sum_{j=1}^{q} 2 \cdot \frac{\text{Precision}_j \cdot \text{Recall}_j}{\text{Precision}_j + \text{Recall}_j}$$

- **Precisión Macro**: Calcula la precisión para cada etiqueta y luego toma la media:

$$\text{Precision (Macro)} = \frac{1}{q} \sum_{j=1}^{q} \frac{\text{TP}_j}{\text{TP}_j + \text{FP}_j}$$

- **Recall Macro**: Calcula el recall para cada etiqueta y luego toma la media:

$$\text{Recall (Macro)} = \frac{1}{q} \sum_{j=1}^{q} \frac{\text{TP}_j}{\text{TP}_j + \text{FN}_j}$$

F1-Score (Samples)

El F1-score por muestras es la media armónica de la precisión y el recall calculados para cada instancia. Es útil cuando se desea evaluar el rendimiento del modelo desde una perspectiva centrada en las instancias.

$$\uparrow \text{F1-Score (Samples)} = \frac{1}{N} \sum_{i=1}^{N} 2 \cdot \frac{\text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}$$

- **Precisión por muestras**:

$$\text{Precision (Samples)} = \frac{1}{N} \sum_{i=1}^{N} \frac{|Y_i \cap \hat{Y}_i|}{|\hat{Y}_i|}$$

- **Recall por muestras**:

$$\text{Recall (Samples)} = \frac{1}{N}\sum_{i=1}^{N}\frac{|Y_i \cap \hat{Y}_i|}{|Y_i|}$$

F1-Score (Micro)

El F1-score micro calcula las métricas agregando todos los verdaderos positivos, falsos positivos y falsos negativos en lugar de promediarlos por etiqueta o por instancia. Es adecuado cuando se desea dar más peso a las etiquetas más frecuentes.

$$\uparrow \text{F1-Score (Micro)} = 2 \cdot \frac{\text{Precision (Micro)} \cdot \text{Recall (Micro)}}{\text{Precision (Micro)} + \text{Recall (Micro)}}$$

- **Precisión micro**:

$$\text{Precision (Micro)} = \frac{\sum_{j=1}^{q} \text{TP}_j}{\sum_{j=1}^{q}(\text{TP}_j + \text{FP}_j)}$$

- **Recall micro**:

$$\text{Recall (Micro)} = \frac{\sum_{j=1}^{q} \text{TP}_j}{\sum_{j=1}^{q}(\text{TP}_j + \text{FN}_j)}$$

En resumen, estas métricas permiten una evaluación exhaustiva del modelo propuesto, proporcionando una visión completa de su rendimiento en diferentes escenarios y contextos.

4.1.3. Ajustes y configuración de los Experimentos

En esta sección se describen las condiciones experimentales empleadas para la evaluación y comparación de los métodos **FCCC**, **CC**, **ECC** y **LCC-MLC**. Con el objetivo de garantizar una comparación justa entre los distintos enfoques, se han definido una serie de configuraciones globales comunes aplicables a todos los algoritmos evaluados.

Configuraciones globales:

1. **Clasificador base:** Se emplea **RandomForestClassifier** de scikit-learn, utilizando un total de **10 árboles** en el bosque aleatorio.
2. **Esquema de validación:** Validación cruzada con **5 particiones** (*5-fold cross-validation*).

3. **Número de experimentos:** Se realizan **30 experimentos** por cada conjunto de datos, combinando las 5 particiones de la validación cruzada con **6 ejecuciones diferentes** utilizando distintas semillas.

Además, dado que algunos de los modelos requieren configuraciones particulares, a continuación se detallan los parámetros específicos utilizados en cada uno de ellos:

FCCC (Fuzzy Clustering Classifier Chains)

1. **Número de *clusters* difusos:** 3.
2. **Umbral de pertenencia:** 0.05.

ECC (Ensemble of Classifier Chains)

1. **Número de cadenas:** 10.

4.2. Ajuste de FCCC

El ajuste del modelo FCCC (Fuzzy Clustering Classifier Chains) implica la configuración de diversos hiperparámetros que afectan directamente tanto a su rendimiento predictivo como a su eficiencia computacional. En esta sección se analizan en detalle los principales parámetros que condicionan su comportamiento: el umbral de pertenencia (*threshold*), el número de clústeres, el orden de la cadena de clústeres y el parámetro de difusidad (m). A través de experimentos sistemáticos, se identifican configuraciones recomendadas en función de distintos escenarios de uso.

4.2.1. Influencia del Threshold en el Rendimiento

El parámetro *threshold* en FCCC controla el umbral mínimo de pertenencia de una etiqueta a un clúster difuso. Este afecta tanto a la precisión como al tiempo de ejecución del modelo. Para entender esta relación se presenta la figura 4.1.

A valores bajos de *threshold*, el modelo permite una mayor superposición entre clústeres, lo que favorece la captura de relaciones complejas entre etiquetas y, por tanto, una mayor precisión; sin embargo, esto incrementa significativamente el coste computacional. Por el contrario, al aumentar el *threshold*, disminuye el número de etiquetas consideradas por clúster, lo que reduce el tiempo de ejecución a costa de una ligera pérdida de precisión.

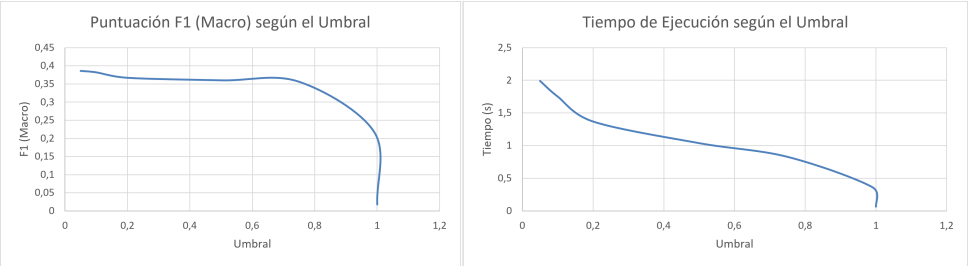

Figura 4.1: F1 Score y tiempo de ejecución promedios en función del umbral.

Para identificar el umbral óptimo, se construyó una métrica compuesta que combina dos factores clave: el rendimiento (F1 Macro) y el coste computacional (tiempo). Como ambas variables tienen escalas distintas, se normalizaron y se definieron tres métricas agregadas:

- **Puntuación equilibrada:** media de F1 normalizado y tiempo normalizado (peso 0.5–0.5).
- **Precisión crítica:** prioriza la precisión (peso 0.75 F1 – 0.25 tiempo).
- **Tiempo crítico:** prioriza la eficiencia (peso 0.25 F1 – 0.75 tiempo).

Los resultados se recogen en la tabla 4.2. Se observa que el mejor equilibrio entre precisión y eficiencia se alcanza con valores de *threshold* entre 0,75 y 0,8.

Tabla 4.2: Evaluación del rendimiento de FCCC en función del *threshold*, considerando tanto precisión como tiempo.

Threshold	F1 (Macro)	Tiempo (s)	Puntuación	Precisión Crítica	Tiempo Crítico
0,05	**0,386**	1,989	1,000	0,750	0,250
0,10	0,382	1,750	1,113	0,773	0,340
0,20	0,367	1,368	1,271	0,792	0,479
0,50	0,360	1,037	1,424	0,821	0,603
0,75	0,356	0,827	**1,522**	**0,840**	0,683
0,99	0,218	**0,362**	1,389	0,619	**0,770**

Recomendaciones prácticas:

- **Entornos con restricciones de tiempo**: Se recomienda un umbral de *threshold*=0,75, ya que ofrece la mejor relación entre tiempo de ejecución y capacidad predictiva. No obstante, es importante evitar valores superiores a 0,75, ya que a partir de ese punto el ahorro en tiempo conlleva una pérdida muy significativa de precisión. Además, acercarse a threshold=1 degrada completamente el modelo ya que se excluyen todas las etiquetas.

- **Sistemas donde la precisión es crítica**: Aunque valores cercanos a 0,75 sigue siendo lo adecuado, si se desea maximizar la precisión sin tener en cuenta el coste computacional, se recomienda usar valores bajos, entre 0,05 y 0,1, especialmente en conjuntos de datos con alta densidad de etiquetas o relaciones complejas.

4.2.2. Análisis del Número Óptimo de Clusters

Se ha realizado un análisis sobre conjuntos de datos con entre 6 y 20 etiquetas para evaluar cómo varían la capacidad predictiva de FCCC y el tiempo de ejecución en función del número de clusters. Dado que el dataset con menor número de etiquetas contiene 6, se ha acotado el número de clusters al rango $[1, 6]$.

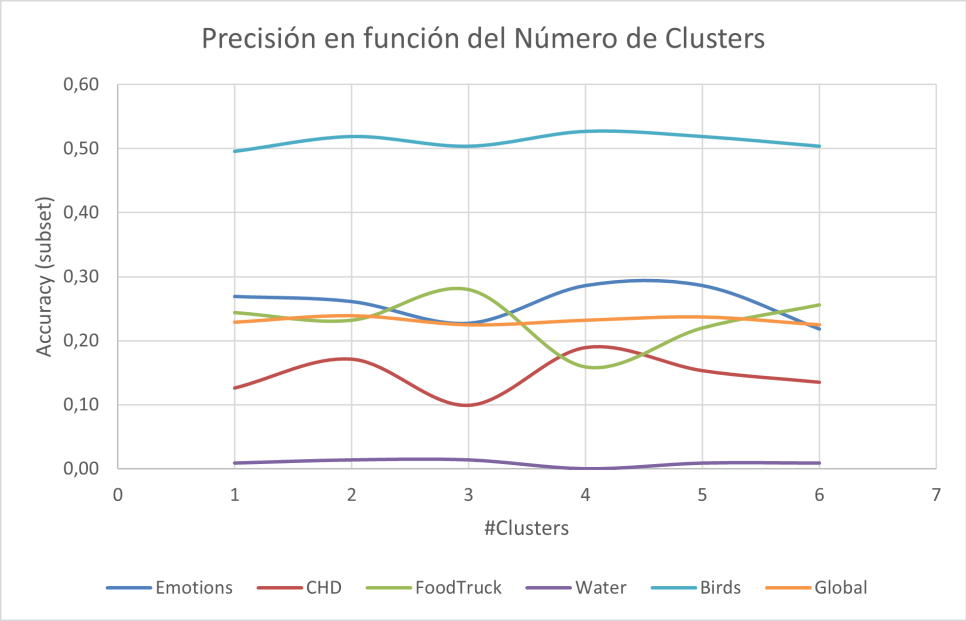

Figura 4.2: Precisión en función del número de clusters para diferentes datasets.

Los resultados experimentales indican que el número óptimo de clusters es altamente dependiente del dataset. No existe un valor universalmente óptimo; la selección debe adaptarse a las características particulares de cada conjunto de datos, buscando un equilibrio entre precisión y eficiencia computacional.

En la Figura 4.2, se observa cómo la precisión varía con el número de clusters para distintos datasets. Cada línea representa un conjunto de datos distinto, y no se identifica un patrón común. Sin embargo, es consistente que las configuraciones extremas (1 o 6 clusters) tienden a ofrecer un rendimiento inferior.

Por otra parte, la Figura 4.3 muestra que el uso de un número intermedio de clusters (en este caso, 3) conlleva un mayor coste computacional. Para esta configuración intermedia se produce el equilibrio entre número de clusters y número de etiquetas por cluster, resultando en una mayor complejidad computacional.

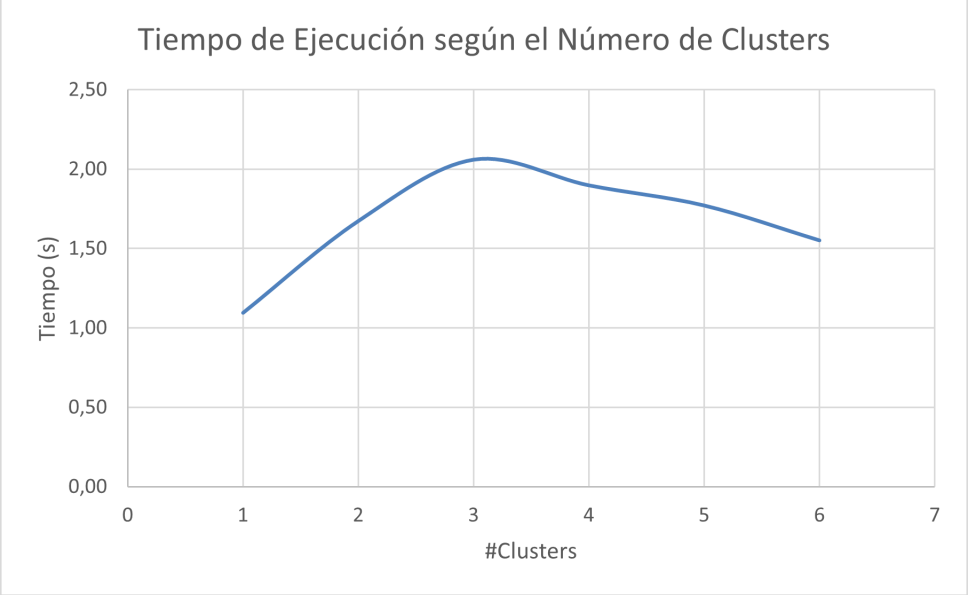

Figura 4.3: Tiempo de ejecución en función del número de clusters para diferentes datasets.

Recomendaciones

- **Elección inicial**: Utilizar 2 clusters como configuración base, ya que proporciona un buen compromiso entre rendimiento y eficiencia.
- **Ajuste fino**: Explorar configuraciones con 2, 3 y 4 clusters, priorizando 3 solo cuando el tiempo de ejecución no sea una restricción crítica.

Conclusiones

1. El número óptimo de clusters depende del dataset, por lo que debe seleccionarse de manera específica para cada caso.
2. El uso de un número intermedio de clusters incrementa el coste computacional debido a la mayor complejidad en la asignación difusa de etiquetas.

4.2.3. Orden de la cadena de clusters

El algoritmo FCCC permite dos estrategias para ordenar los clústeres durante la construcción de la cadena de clasificadores:

- **Orden aleatorio**: Permutación uniforme de los índices de los clústeres
- **Orden heurístico**: Secuencia basada en dependencias entre clústeres (de menos a más dependientes).

La Tabla 4.3 muestra el impacto de ambas estrategias:

Tabla 4.3: Mejora de la estrategia heurística respecto a la aleatoria en métricas de evaluación

Métrica	Aleatorio	Heurístico	Mejora %
F1 Macro	0.155	0.386	149.71 %
F1 Micro	0.235	0.539	128.83 %
F1 Samples	0.175	0.454	159.37 %
Accuracy	0.144	0.225	56.07 %
Hamming Loss	0.242	0.203	16.46 %
Tiempo (s)	3.99	2.02	49.34 %

Los hallazgos clave son:

- **Eficiencia de propagación**: La ordenación heurística optimiza el flujo de información entre clústeres, reduciendo un 49.34 % el tiempo de ejecución respecto al orden aleatorio.
- **Calidad predictiva**: Todas las métricas mejoran significativamente:
  - **F1 Samples** (159.37 %): Mayor beneficio en predicción por instancia
  - **Hamming Loss** (16.46 %): Menos errores en etiquetas individuales

  Conclusión: La estrategia heurística demuestra ser superior tanto teórica como empíricamente. Al procesar primero los clústeres menos dependientes y propagar gradualmente su información a los más dependientes, se optimiza tanto la precisión como la eficiencia computacional del modelo.

4.2.4. Análisis del Parámetro de Difusidad (m)

El parámetro m en Fuzzy C-Means controla el grado de solapamiento entre clusters, donde valores mayores permiten una pertenencia más difusa a múltiples clusters. Como muestran las Figuras 4.4 y 4.5, este parámetro afecta significativamente tanto al rendimiento como a la eficiencia del modelo.

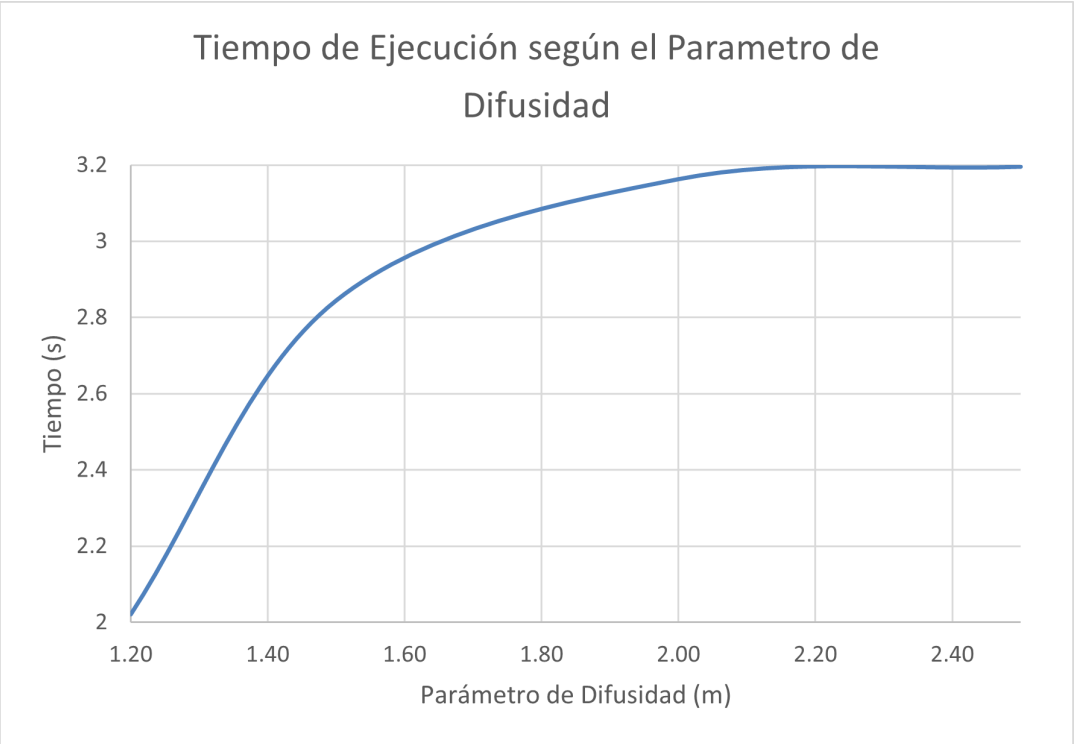

Figura 4.4: Tiempo de ejecución en función del parámetro de difusidad m

Hallazgos clave

- **Rendimiento predictivo**:
  - El F1 Macro alcanza su máximo en $m = 2{,}0$ (0.413) con una mejora del 7 % respecto a $m = 1{,}2$
  - La Accuracy muestra su pico en $m = 2{,}0$ (0.243), confirmando este valor como óptimo
  - Hamming Loss también alcanza su mínimo (0.202) alrededor de $m = 2{,}0$, como evidencia la Figura 4.5
  - Las métricas F1 Micro y Samples siguen patrones similares con mejoras del 4-5 %
- **Eficiencia computacional** (Figura 4.4):
  - Tiempo de ejecución crece linealmente con m (58 % más lento en $m = 2{,}5$ vs $m = 1{,}2$)
  - El incremento se debe a mayores cálculos en la actualización de grados de pertenencia
- **Compromiso óptimo**: $m = 2{,}0 \pm 0{,}3$ ofrece el mejor balance entre precisión y eficiencia

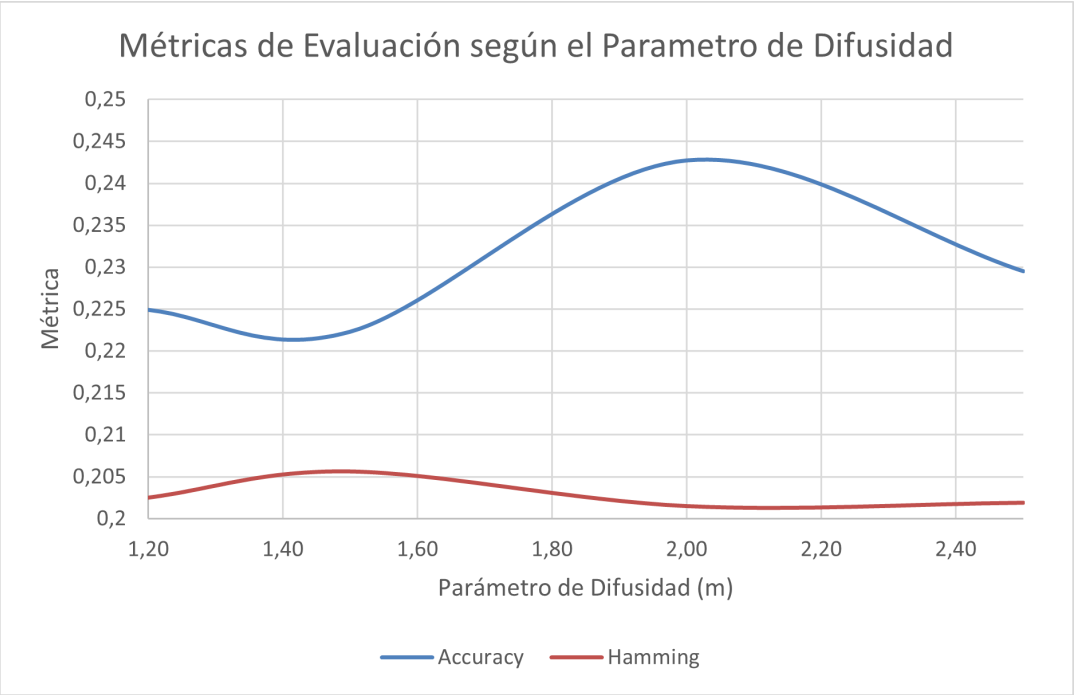

Figura 4.5: Accuracy y Hamming Loss en función de m

4.3. Comparación con el Estado del Arte

4.3.1. Resultados Experimentales

En esta sección se presentan los resultados obtenidos por los diferentes algoritmos evaluados. Las Tablas 4.4-4.8 muestran los valores promedio de cada métrica para los conjuntos de datos analizados, donde los mejores resultados para cada conjunto de datos aparecen resaltados en negrita.

4.3.2. Análisis de Resultados

Para evaluar objetivamente el rendimiento del algoritmo FCCC en relación con otros métodos del estado del arte (CC, ECC y LCC-MLC), se ha realizado un estudio cuantitativo basado en la *proximidad relativa* de sus resultados. Esta medida expresa, en porcentaje, cuán cerca (o por encima) está FCCC respecto a cada uno de los métodos de comparación para distintas métricas de evaluación.La fórmula empleada para calcular esta medida es:

$$\text{Proximidad Relativa}(FCCC, X) = \begin{cases} 100 \times \frac{FCCC}{X}, & \text{si la métrica se maximiza} \\ 100 \times \frac{X}{FCCC}, & \text{si la métrica se minimiza} \end{cases}$$

Tabla 4.4: Resultados de ↑ **F1-Score (Macro)** por conjunto de datos

Dataset	**CC**	**ECC**	**FCCC**	**LCC-MLC**
Emotions	0.606	**0.653**	0.604	0.599
CHD_49	0.457	**0.509**	0.444	0.445
VirusGo	0.779	**0.794**	0.776	0.763
Foodtruck	0.158	**0.190**	0.162	0.151
Water-quality	0.456	**0.526**	0.476	0.446
Birds	0.188	**0.242**	0.237	0.173
Yahoo_Arts	0.163	0.204	**0.365**	0.313
Yahoo_Business	0.167	0.198	**0.203**	0.191
Mediamill	0.155	0.193	**0.220**	0.175
Bibtex	0.085	0.107	**0.110**	0.101
Promedio	0.321	**0.362**	0.360	0.336

Tabla 4.5: Resultados de ↑ **F1-Score (Micro)** por conjunto de datos

Dataset	**CC**	**ECC**	**FCCC**	**LCC-MLC**
Emotions	0.628	**0.673**	0.623	0.627
CHD_49	0.616	**0.664**	0.615	0.621
VirusGo	0.842	**0.858**	0.847	0.844
Foodtruck	0.470	**0.515**	0.463	0.464
Water-quality	0.523	**0.584**	0.534	0.519
Birds	0.354	**0.437**	0.424	0.334
Yahoo_Arts	0.342	0.407	**0.533**	0.457
Yahoo_Business	0.732	0.749	**0.757**	0.745
Mediamill	0.582	**0.621**	0.613	0.591
Bibtex	0.256	0.287	**0.291**	0.276
Promedio	0.534	**0.580**	0.570	0.548

Donde X representa uno de los métodos de comparación (CC, ECC o LCC-MLC), y los valores corresponden a los resultados medios obtenidos por cada algoritmo en las métricas evaluadas. De esta forma:

- Un valor de **100 %** indica un rendimiento idéntico al algoritmo comparado.
- Un valor **mayor a 100 %** indica que FCCC supera en rendimiento al algoritmo comparado.
- Un valor **menor a 100 %** indica que FCCC rinde por debajo del comparado.

En la Tabla 4.10 se presentan los valores de proximidad relativa de FCCC respecto a los tres algoritmos considerados, para cada métrica. La última columna muestra el promedio de FCCC respecto a los tres algoritmos en esa métrica, y la última fila resume la media de cada columna, es decir, el rendimiento general promedio de FCCC frente a cada método.

Los resultados muestran que FCCC presenta un rendimiento global muy

Tabla 4.6: Resultados de ↑ **F1-Score (Samples)** por conjunto de datos

Dataset	**CC**	**ECC**	**FCCC**	**LCC-MLC**
Emotions	0.566	**0.634**	0.549	0.563
CHD_49	0.582	**0.640**	0.574	0.587
VirusGo	0.829	**0.857**	0.835	0.834
Foodtruck	0.479	**0.529**	0.469	0.475
Water-quality	0.485	**0.553**	0.496	0.483
Birds	0.134	**0.174**	0.167	0.122
Yahoo_Arts	0.287	0.360	**0.495**	0.406
Yahoo_Business	0.770	0.790	**0.791**	0.788
Mediamill	0.543	**0.590**	0.578	0.557
Bibtex	0.216	0.244	**0.247**	0.237
Promedio	0.489	**0.537**	0.520	0.505

Tabla 4.7: Resultados de ↑ **Accuracy (Subset)** por conjunto de datos

Dataset	**CC**	**ECC**	**FCCC**	**LCC-MLC**
Emotions	0.262	**0.291**	0.259	0.263
CHD_49	**0.145**	0.137	0.131	0.132
VirusGo	0.763	**0.773**	0.765	0.766
Foodtruck	**0.231**	0.226	0.211	0.225
Water-quality	0.017	**0.018**	0.016	0.016
Birds	0.506	**0.523**	0.517	0.504
Yahoo_Arts	0.209	0.252	**0.400**	0.341
Yahoo_Business	0.578	0.587	**0.598**	0.591
Mediamill	0.160	**0.174**	0.163	0.153
Bibtex	0.104	0.119	**0.120**	0.117
Promedio	0.297	0.310	**0.318**	0.311

competitivo frente a los métodos del estado del arte. En la mayoría de las métricas evaluadas, FCCC supera a al menos dos de los tres algoritmos comparados. En conjunto, FCCC alcanza una media general del **103.15 %**, lo que representa una mejora promedio del 3.15 % respecto al conjunto de métodos comparados. Estos resultados refuerzan la validez y solidez del enfoque propuesto.

4.3.3. Tests Estadísticos

Para comparar el rendimiento de los algoritmos propuestos, se han realizado tests estadísticos sobre las métricas obtenidas. En particular, se ha empleado el **test de Friedman** para detectar diferencias significativas entre métodos, seguido del **test de Nemenyi** como análisis post-hoc en los casos necesarios. Ambos tests han sido implementados mediante la librería `Statds` [13].

Tabla 4.8: Resultados de ↓ **Hamming Loss** por conjunto de datos

Dataset	**CC**	**ECC**	**FCCC**	**LCC-MLC**
Emotions	0.206	**0.200**	0.204	0.206
CHD_49	0.318	0.315	0.312	**0.310**
VirusGo	0.061	**0.057**	0.059	0.060
Foodtruck	0.155	**0.154**	0.157	0.157
Water-quality	0.291	**0.289**	0.294	0.293
Birds	0.044	**0.042**	0.042	0.045
Yahoo_Arts	0.055	0.055	**0.044**	0.045
Yahoo_Business	0.025	0.024	**0.023**	0.024
Mediamill	0.028	**0.027**	0.028	0.028
Bibtex	0.013	0.013	**0.013**	0.013
Promedio	0.120	**0.117**	0.118	0.118

Tabla 4.9: Resultados de ↓ **Tiempo de Ejecución (s)** por conjunto de datos

Dataset	**CC**	**ECC**	**FCCC**	**LCC-MLC**
Emotions	0.42	0.82	0.61	**0.13**
CHD_49	0.36	0.68	0.59	**0.13**
VirusGo	0.43	0.73	0.60	**0.36**
Foodtruck	0.73	1.38	1.42	**0.45**
Water-quality	0.95	1.88	3.18	**0.15**
Birds	1.28	2.56	5.77	**0.64**
Yahoo_Arts	97.80	214.46	311.90	**53.87**
Yahoo_Business	150.72	285.38	983.27	**85.89**
Mediamill	86.56	176.40	241.10	**35.21**
Bibtex	35.70	65.60	150.71	**5.34**
Promedio	37.495	74.989	169.915	**18.217**

Test de Friedman

La Tabla 4.11 muestra los resultados del test de Friedman para todas las métricas evaluadas. Se incluyen los p-valores obtenidos y si se rechaza o no la hipótesis nula de que todos los algoritmos tienen el mismo rendimiento estadístico.

Tabla 4.10: Proximidad relativa (%) de FCCC respecto a otros métodos

Métrica	**CC**	**ECC**	**LCC-MLC**	**General**
F1 Macro	111.95	99.49	107.09	106.18
F1 Micro	106.67	98.37	104.05	103.03
F1 Samples	106.29	96.85	102.94	102.03
Accuracy	106.92	102.59	102.38	103.97
Hamming Loss	101.54	99.81	100.27	100.54
General	**106.07**	**99.42**	**103.75**	**103.15**

Tabla 4.11: Resultados del test de Friedman para cada métrica.

Métrica	**p-valor**	**¿Rechazo de H_0?**
F1-Macro	0.002	Sí
F1-Micro	0.004	Sí
F1-Samples	0.004	Sí
Subset Accuracy	0.136	No
Hamming Loss	0.043	Sí
Tiempo ejecución	0.001	Sí

Como se observa, existen diferencias significativas entre algoritmos para todas las métricas, salvo en *Subset Accuracy*, donde no se puede rechazar la hipótesis nula.

Test de Nemenyi

En aquellas métricas donde el test de Friedman detectó diferencias significativas, se ha aplicado el **test de Nemenyi** para realizar comparaciones múltiples y detectar qué métodos difieren estadísticamente entre sí.

A continuación, se presentan los diagramas de Nemenyi correspondientes a las métricas relevantes. Cuando dos algoritmos están unidos por una línea horizontal, esto indicaría que no es posible afirmar que su rendimiento sea significativamente distinto. Cada figura muestra los rankings medios de los algoritmos junto con el valor crítico de diferencia (CD).

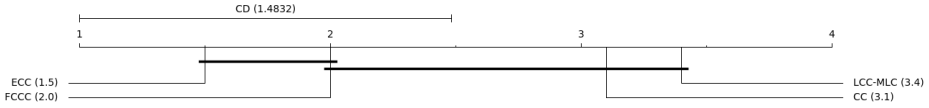

Figura 4.6: Diagrama de Nemenyi para F1-Macro.

Figura 4.7: Diagrama de Nemenyi para F1-Micro.

Figura 4.8: Diagrama de Nemenyi para F1-Samples.

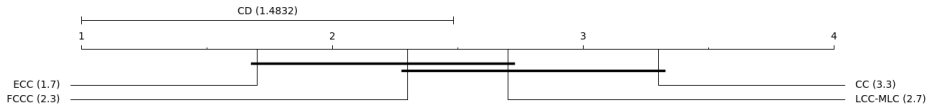

Figura 4.9: Diagrama de Nemenyi para Hamming Loss.

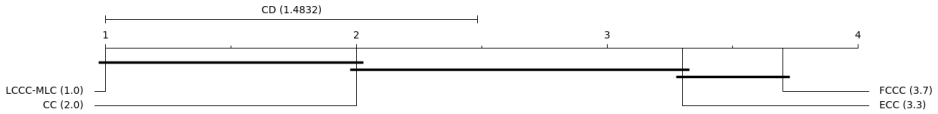

Figura 4.10: Diagrama de Nemenyi para tiempo de ejecución.

Discusión de los resultados

- **Precisión**: ECC destaca especialmente en F1-Score (en todas sus variantes), mientras que en subset accuracy no exite una diferencia significativa entre el rendimiento de los algoritmos estudiados.
- **Eficiencia**: LCC-MLC y CC son los algoritmos más eficientes en términos de tiempo de ejecución, con diferencias muy significativas frente a FCCC.
- **Consistencia**: FCCC se posiciona de manera consistente como el segundo mejor algoritmo en todas las métricas, excepto en el tiempo de ejecución, mostrando un rendimiento similar al de ECC, que lidera en la mayoría de los casos.

Dados estos resultados, podemos concluir que FCCC es una propuesta atractiva, ya que mejora el rendimiento predictivo de LCC-MLC, un algoritmo con características similares que agrupa las etiquetas en pequeños clústeres rígidos (en lugar de difusos) antes de aplicar el algoritmo CC. Por otro lado, es destacable el buen rendimiento de FCCC, que en ninguna métrica tiene un rendimiento significativamente distinto al mejor algoritmo, que es ECC en todos los casos; sin embargo, este mismo hecho también revela que FCCC no logra superarlo, a pesar de tener una complejidad ligeramente superior.

En este sentido, FCCC se posiciona como una propuesta prometedora, con resultados competitivos y una capacidad demostrada para modelar relaciones complejas entre etiquetas. No obstante, también se identifica un margen de mejora, lo que abre oportunidades interesantes para trabajos futuros.

4.3.4. Análisis del tiempo de Ejecución y Escalabilidad

Además del rendimiento predictivo, se ha evaluado el coste computacional de los algoritmos en función de la complejidad de los conjuntos de datos. Como métrica de complejidad se ha utilizado $\log_{10}(m \cdot d \cdot q)$, donde m es el número de instancias, d el número de atributos y q el número de etiquetas.

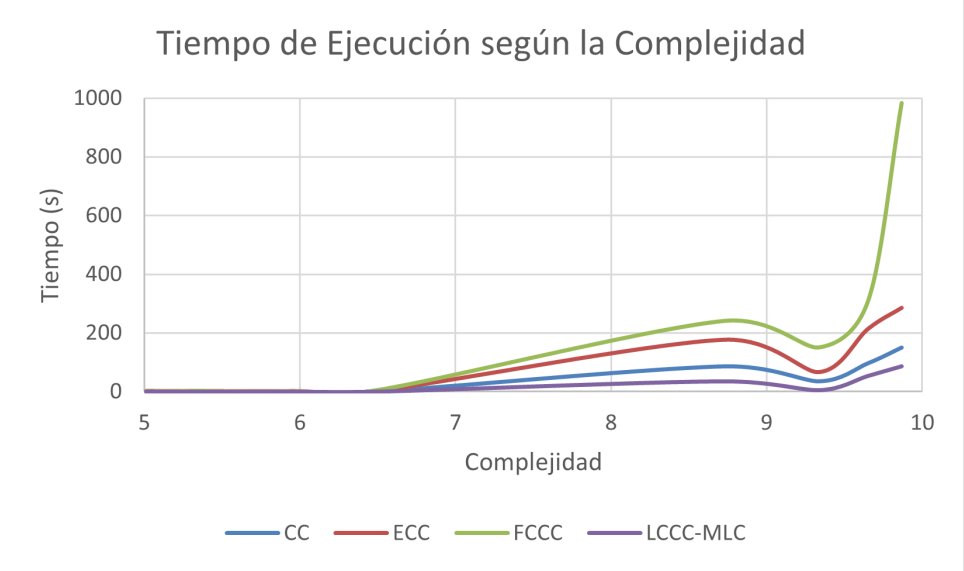

Figura 4.11: Tiempos de ejecución de los algoritmos en función de la complejidad del conjunto de datos ($\log_{10}(m \cdot d \cdot q)$).

En el caso de **FCCC**, el comportamiento es dual: para problemas de **baja a media complejidad** (log-complejidad menor o igual a 6), el tiempo de

ejecución es comparable al de CC y ECC. Sin embargo, a partir de niveles de **alta complejidad** (log-complejidad superior a 8), FCCC muestra una escalada abrupta del tiempo computacional, siendo el más lento con diferencia. En el caso más extremo, supera en más de 9 veces el tiempo de ejecución de ECC y en más de 10 veces el de LCC-MLC.

En consecuencia, se concluye que FCCC es una alternativa eficaz para tareas de clasificación multi-etiqueta en entornos de baja o media escala, donde su mayor capacidad predictiva compensa su coste computacional. Sin embargo, en escenarios de alta complejidad, su uso debe ser evaluado con cautela, especialmente si se requiere una ejecución eficiente o en tiempo real.

4.3.5. Rendimiento de FCCC en Datasets de Alta Complejidad

Dado que se observa un considerable incremento en el tiempo de ejecución en *FCCC* para los datasets complejos, es necesario analizar si esta desventaja se compensa con una mejora en la precisión de las predicciones.

Para ello, en la Figura 4.12 se presenta la puntuación F1 Macro obtenida por cada algoritmo en función de la complejidad del conjunto de datos, medida como $\log_{10}(m \cdot d \cdot q)$. A partir de una complejidad cercana a 8, se puede observar que *FCCC* comienza a destacar como el modelo con mayor precisión.

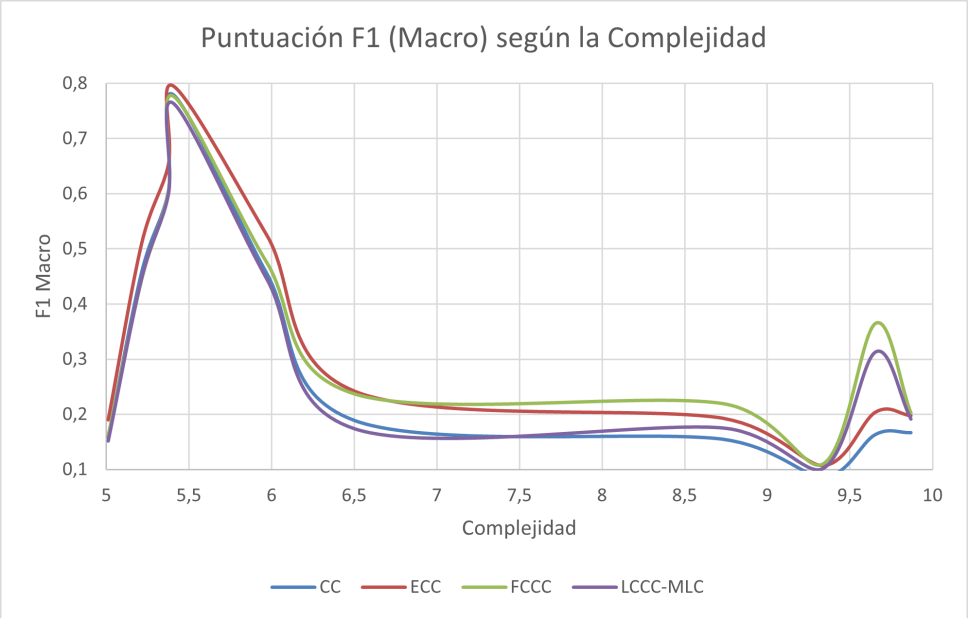

Figura 4.12: F1 Score en función de la complejidad del conjunto de datos ($\log_{10}(m \cdot d \cdot q)$).

Considerando como *datasets complejos* aquellos cuya complejidad es mayor a 8 —es decir, **Mediamill**, **Bibtex**, **Yahoo_Arts** y **Yahoo_Business**—, se ha calculado el F1-score promedio para cada algoritmo. Además, se ha cuantificado la mejora relativa de FCCC respecto a los otros tres métodos mediante la siguiente fórmula:

$$\text{Mejora}(FCCC, X) = \frac{FCCC - X}{X} \times 100$$

donde X representa el F1-score medio del algoritmo comparado. Los resultados se muestran en la Tabla 4.12.

Dataset	**Complexity**	**CC**	**ECC**	**FCCC**	**LCCC-MLC**
Mediamill	8.73	0.15	0.19	0.22	0.18
Bibtex	9.33	0.09	0.11	0.11	0.10
Yahoo_Arts	9.65	0.16	0.20	0.37	0.31
Yahoo_Business	9.87	0.17	0.20	0.20	0.19
Media	–	0.14	0.18	0.22	0.20
Mejora porcentual	–	57.56	27.92	–	15.02

Tabla 4.12: Estudio de la puntuación F1 Macro en datasets complejos donde se analiza la mejora de FCCC respecto al resto de algoritmos.

Como se observa en la Tabla 4.12, FCCC supera en promedio al resto de algoritmos en precisión para datasets complejos: un **57,6 %** de mejora respecto a *CC*, un **27,9 %** frente a *ECC* y un **15,0 %** sobre *LCCC-MLC*.

Estos datos confirman que, a pesar del mayor coste computacional, FCCC se posiciona como el modelo más efectivo cuando se trata de estructuras de etiquetas densas o extensas, justificando su uso en escenarios de alta complejidad.

5. Conclusiones y Trabajo Futuro

5.1. Conclusiones

En este Trabajo Fin de Grado se ha propuesto el algoritmo Fuzzy Cluster-Based Classifier Chain (FCCC), una nueva aproximación para la clasificación multi-etiqueta que integra técnicas de clustering difuso con el modelo de cadenas de clasificadores. A diferencia de los enfoques tradicionales como Classifier Chains (CC) o Label Clusters Chains Multi-Label Classifier (LCC-MLC), FCCC permite que una etiqueta pertenezca a múltiples clústeres con diferentes grados de pertenencia, capturando así relaciones más complejas entre etiquetas y mejorando la capacidad predictiva del modelo.

Los resultados experimentales, realizados sobre 10 conjuntos de datos de referencia que abarcan diversos dominios, demuestran que FCCC es una alternativa competitiva frente a los métodos del estado del arte. El análisis estadístico mediante tests de Friedman y Nemenyi confirmó que FCCC no presenta diferencias significativas con ECC en métricas clave como F1 Macro, F1 Micro y F1 Samples, posicionándose consistentemente como segundo mejor en la mayoría de métricas, siendo ECC el que ocupa el primer lugar. En cuanto a eficiencia, los tests estadísticos revelaron que LCC-MLC y CC son los más rápidos, con diferencias significativas frente a FCCC.

La capacidad predictiva de FCCC sobresale especialmente en conjuntos de datos de alta complejidad, donde FCCC superó en un 57.6 % a CC, un 27.9 % a ECC y un 15.0 % a LCC-MLC en términos de F1-Score Macro, demostrando su capacidad para modelar relaciones complejas entre etiquetas. A cambio de este aumento en la precisión, se observa que el tiempo de ejecución de FCCC puede llegar a ser hasta 5 veces superior al de ECC.

El estudio de los hiperparámetros de FCCC reveló que el rendimiento óptimo se alcanza con un parámetro de difusidad $m = 2{,}0$, un umbral de pertenencia entre 0.75 y 0.8 para maximizar la precisión y minimizar el tiempo de ejecución. Asimismo, la estrategia de ordenación heurística

de clústeres demostró ser claramente superior a la ordenación aleatoria, mejorando tanto la precisión (hasta un 159.37 % en F1 Samples) como la eficiencia computacional (reducción del 49.34 % en tiempo de ejecución).

A nivel personal, la realización de este trabajo ha supuesto un profundo aprendizaje tanto teórico como práctico. He podido comprender en profundidad los desafíos específicos de la clasificación multi-etiqueta y las limitaciones de los enfoques tradicionales. El desarrollo de FCCC me ha permitido aplicar conocimientos de diversas áreas como aprendizaje supervisado y automático, teoría de conjuntos difusos y análisis estadístico, integrándolos en una solución coherente y efectiva. Además, he adquirido habilidades en el diseño y ejecución de experimentos y análisis de resultados. Este trabajo no solo ha ampliado mi comprensión técnica, sino que también me ha enseñado la importancia de abordar problemas complejos desde múltiples perspectivas, buscando soluciones que equilibren precisión, eficiencia computacional y aplicabilidad práctica.

5.2. Limitaciones y Trabajo Futuro

A partir de los resultados obtenidos en este trabajo, se han identificado varias limitaciones del algoritmo FCCC que abren interesantes líneas de investigación futura:

5.2.1. Limitaciones

Eficiencia computacional en datasets complejos: Los tests estadísticos han demostrado que LCC-MLC y CC son significativamente más rápidos que FCCC. Esta diferencia se acentúa especialmente en datasets de alta complejidad (log-complejidad > 8), donde el tiempo de ejecución de FCCC puede multiplicarse hasta por diez respecto a otros métodos. Esta limitación restringe su aplicabilidad en entornos donde el tiempo de respuesta es crítico o se manejan grandes volúmenes de datos.

Rendimiento predictivo respecto a ECC: Aunque FCCC se posiciona consistentemente como el segundo mejor algoritmo en la mayoría de métricas evaluadas, los tests estadísticos confirman que ECC sigue siendo superior en términos de precisión (F1 Macro, F1 Micro y F1 Samples). Esta brecha, aunque no estadísticamente significativa, sugiere que existe margen de mejora en la capacidad predictiva de FCCC.

Sensibilidad a hiperparámetros: El rendimiento de FCCC depende considerablemente de la correcta configuración de varios hiperparámetros, como el número de clústeres, el umbral de pertenencia y el parámetro de difusidad m. Esta dependencia puede dificultar su aplicación.

Equilibrio entre precisión y eficiencia: Los experimentos muestran que maximizar la precisión de FCCC (con umbrales bajos de 0.05-0.1) implica un mayor coste computacional. Este compromiso entre precisión y eficiencia puede resultar problemático en aplicaciones con restricciones de recursos.

5.2.2. Trabajo Futuro

Optimización del rendimiento computacional: Desarrollar implementaciones paralelas o distribuidas de FCCC utilizando frameworks como Apache Spark o TensorFlow, especialmente orientadas a mejorar el rendimiento en datasets de alta dimensionalidad. También se podría explorar la aplicación de técnicas de reducción de dimensionalidad específicas para entornos multi-etiqueta como paso previo al clustering difuso.

Integración con técnicas de ensemble: Dado el buen rendimiento de ECC, una línea prometedora sería desarrollar un modelo híbrido que combine las fortalezas de FCCC (modelado de relaciones complejas entre etiquetas) con las de ECC (robustez mediante ensemble). Esto podría implementarse mediante técnicas de stacking o mediante la integración de múltiples cadenas de clústeres difusos con diferentes configuraciones.

Autoajuste de hiperparámetros: Implementar mecanismos de optimización automática que determinen los valores óptimos de hiperparámetros (número de clústeres, umbral, parámetro m) en función de las características del dataset. Técnicas como Bayesian Optimization podrían emplearse para este fin, reduciendo la necesidad de ajuste manual.

Exploración de otras técnicas de clustering difuso: Investigar alternativas al algoritmo Fuzzy C-Means que puedan ofrecer mejor rendimiento o escalabilidad, como Possibilistic C-Means, Gustafson-Kessel o algoritmos de clustering difuso jerárquico. Estas variantes podrían adaptarse mejor a la estructura específica de las relaciones entre etiquetas.

Adaptación a dominios específicos: Desarrollar versiones especializadas de FCCC para dominios particulares como clasificación de texto, imágenes o datos biomédicos, incorporando conocimiento específico del dominio en la fase de clustering. Por ejemplo, en clasificación de texto, se podrían utilizar embeddings semánticos para mejorar la formación de clústeres.

La implementación de estas mejoras podría potenciar significativamente el algoritmo FCCC, consolidándolo como una alternativa competitiva y versátil para problemas de clasificación multi-etiqueta en diversos dominios de aplicación.

Bibliografía

[1] F. C. F. Herrera. Multilabel classification: Problem analysis, metrics and techniques. *Springer*, 2016.

[2] K. Dembczynski, W. Waegeman, W. Cheng, and E. Hüllermeier. On label dependence in multi-label classification. In *Workshop Proceedings of Learning from Multi-Label Data*, pages 5–12, 2010.

[3] J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. In W. Buntine, M. Grobelnik, D. Mladenic, and J. Shawe-Taylor, editors, *Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2009*, volume 5782 of *Lecture Notes in Computer Science*, pages 254–269. Springer, Berlin, Heidelberg, 2009.

[4] L. Rokach. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2), 2010.

[5] Elaine Cecília Gatto, Mauri Ferrandin, and Ricardo Cerri. Multi-label classification with label clusters. *Knowledge and Information Systems*, 67(2):1741–1785, 2025.

[6] Jesse Read and Fernando Perez-Cruz. Deep learning for multi-label classification. *arXiv preprint*, 2014.

[7] Eva Gibaja and Sebastián Ventura. Multi-label learning: a review of the state of the art and ongoing research. *WIREs Data Mining and Knowledge Discovery*, 4(6):411–444, 2014.

[8] X. Ran, Y. Xi, Y. Lu, X. Wang, and Z. Lu. Comprehensive survey on hierarchical clustering algorithms and the recent developments. *Artificial Intelligence Review*, 222, 2022.

[9] K. R. Shahapure and C. Nicholas. Cluster quality analysis using silhouette score. In *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*, 2020.

[10] L. A. Zadeh. Fuzzy sets. *Information and Control*, pages 338–353, 1965.

[11] J. C. Bezdek, R. Ehrlich, and W. Full. Fcm: The fuzzy c-means clustering algorithm. *Computers Geosciences*, 10(2-3):191–203, 1984.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[13] Christian Luna, Antonio R Moya, José María Luna, and Sebastián Ventura. Statds library: statistical tests for data science. *Neurocomputing*, 595:127877, 2024.