Clasificación de texto jerarárquica 2024-2025

Master Ciencia de Datos

Universidad de Granada

Clasificación Jerárquica de Textos

Miguel García López

Índice

Índice de figuras

Índice de cuadros

1. Introducción

La clasificación de textos es una de las tareas fundamentales en el procesamiento del lenguaje natural (PLN), permitiendo la asignación automática de etiquetas a documentos en función de su contenido. Tradicionalmente, los enfoques de clasificación han tratado las etiquetas de forma independiente, aplicando modelos *flat* que ignoran las relaciones inherentes entre ellas. Sin embargo, en muchos contextos reales, las etiquetas están organizadas de forma jerárquica, lo que implica relaciones de dependencia y estructuras taxonómicas complejas.

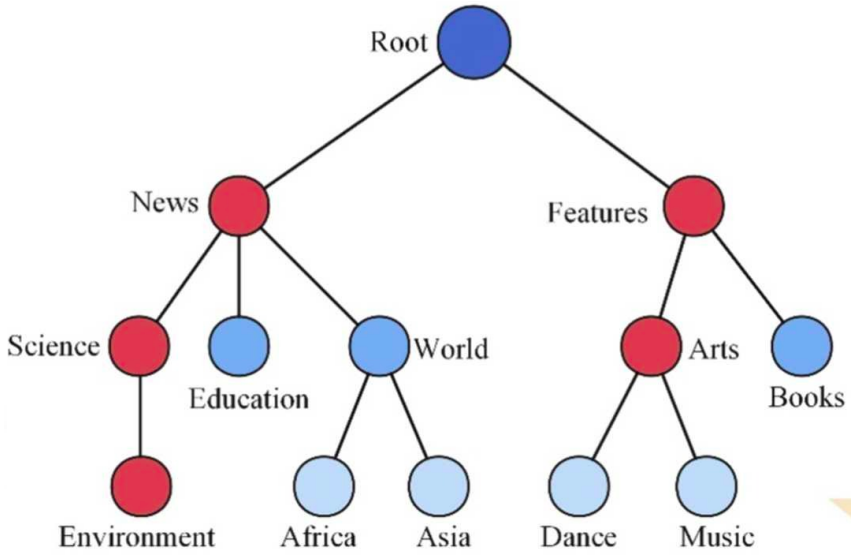


Figura 1: Ejemplo de clasificación jerarárquica de texto.

La **clasificación jerárquica de textos** (HTC, por sus siglas en inglés) surge para abordar esta limitación, aprovechando la estructura de árbol o de grafo que organiza las etiquetas. Este enfoque no solo permite capturar la relación entre categorías padre e hijo, sino que también facilita la generalización de modelos frente a nuevas clases derivadas de categorías pre-existentes.

En este documento se experimenta con tecnologías como **PyTorch** para abordar este problemas y ahondar en la arquitectura de redes como *LSTM* para la clasificación de texto. Además se implementan arquitecturas multi-modelo específicamente propuestos para abordar este tipo de problemas jerárquicos.

2. Estructura del problema

Sea $D = \{d_1, d_2, \ldots, d_n\}$ un conjunto de documentos de texto, donde cada d_i representa un documento individual.

Sea $L = \{l_1, l_2, \ldots, l_m\}$ un conjunto de etiquetas organizadas en una estructura jerárquica, que puede representarse como un árbol o un grafo.

La relación jerárquica entre las etiquetas se define mediante una función $H : L \times L \to \{0, 1\}$, donde:

$$H(l_i, l_j) = \begin{cases} 1, & \text{si } l_i \text{ es un ancestro de } l_j \text{ en la jerarquía,} \\ 0, & \text{en caso contrario.} \end{cases}$$

El objetivo es encontrar una función de clasificación que asigne a cada documento d_i un conjunto de etiquetas $L_i \subseteq L$, respetando la jerarquía de etiquetas. Es decir, no se puede clasificar un documento d_i como "categoría 2" si esta no es hija de "categoria 1" y previamente ese documento fue clasificado con esa etiqueta, es una discrepancia estructural con el grafo de jerarquía.

3. Tipos de clasificadores

3.1. Flat

En este enfoque se ignora la jerarquía y se trata cada etiqueta de forma independiente, reduciendo el problema a una clasificación tradicional (ya sea multicategoría o *multilabel*). Normalmente, solo se consideran los nodos hoja, de modo que se pierde la información estructural de la jerarquía.

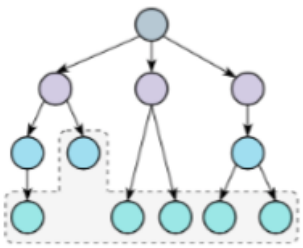


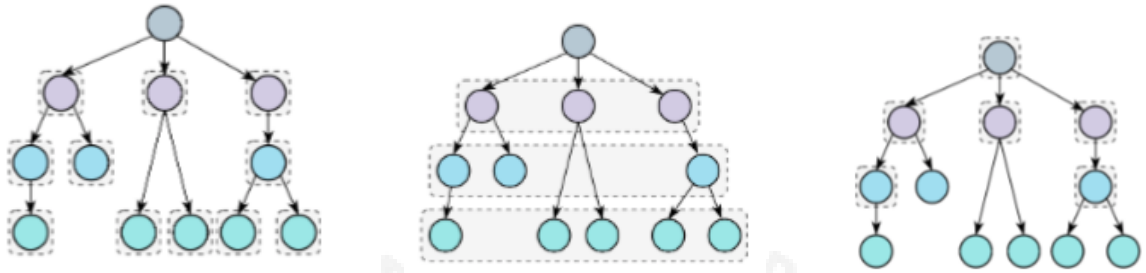
Figura 2: Arquitectura de clasificadores plana.

3.2. Local Classifiers

Estos métodos dividen el problema en subproblemas que se ajustan a la estructura jerárquica. Se pueden identificar tres subtipos:

- **Local por Nodo:** Se entrena un clasificador binario para cada nodo de la jerarquía, tratando cada etiqueta de forma individual.

- **Local por Nivel:** Se entrena un clasificador para cada nivel de la jerarquía, aprovechando la información compartida por las etiquetas del mismo nivel.
- **Local por Padre:** Cada clasificador asociado a un nodo padre decide entre sus hijos, permitiendo capturar la dependencia directa entre categorías.



(a) Clasificación por nodo. (b) Clasificación por nivel. (c) Clasificación por padre.

Figura 3: Diferentes arquitecturas de clasificación jerárquica.

3.3. Global Classifiers

En este enfoque se utiliza un único modelo que integra de forma global la estructura jerárquica de las etiquetas. Este modelo considera toda la información estructural durante el proceso de entrenamiento y la inferencia, lo que puede resultar en una mejor generalización y coherencia en las predicciones.

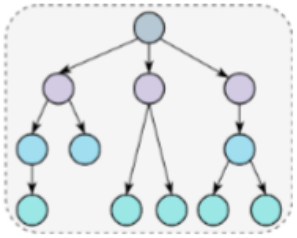


Figura 4: Clasificador global.

4. Dataset

El conjunto de datos **Hierarchical Text Classification**, disponible en *Kaggle*, proporciona información estructurada para la clasificación jerárquica de textos, basada en reseñas de productos de *Amazon*. Su objetivo es facilitar el desarrollo y la comparación de métodos de clasificación de texto que aprovechen la estructura jerárquica de las etiquetas.

Este dataset contiene reseñas de productos de Amazon organizadas en una jerarquía de tres niveles de categorías:

- **Nivel 1:** 6 categorías principales (ej. salud y cuidado personal, juguetes y juegos, belleza, entre otros).
- **Nivel 2:** 64 subcategorías derivadas de las clases principales.
- **Nivel 3:** 510 categorías más específicas dentro de cada subcategoría.

Se presentan tres archivos de datos:

- **train_40k.csv:** Conjunto de entrenamiento con 40,000 reseñas de productos.
- **valid_10k.csv:** Conjunto de validación con 10,000 reseñas.

Cada fila del conjunto de datos contiene:

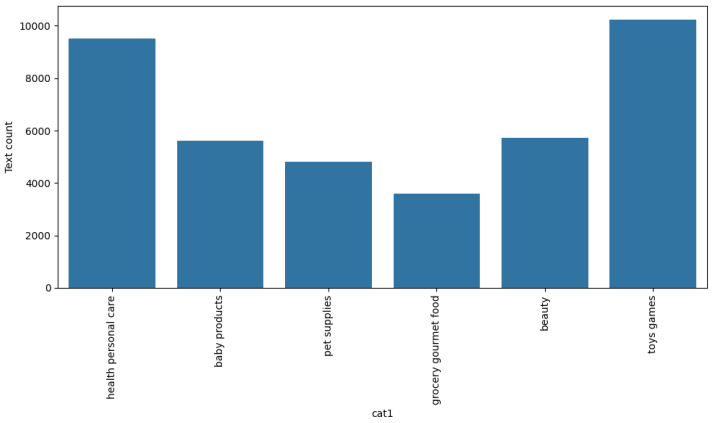
- ID del producto.
- Título de la reseña.
- ID del usuario.
- Información sobre la utilidad de la reseña.
- Puntaje de calificación otorgado por otros usuarios.
- Fecha de la reseña.
- Texto completo de la reseña.
- Etiquetas de clasificación para los niveles 1, 2 y 3.

4.1. Posibles Enfoques de Clasificación

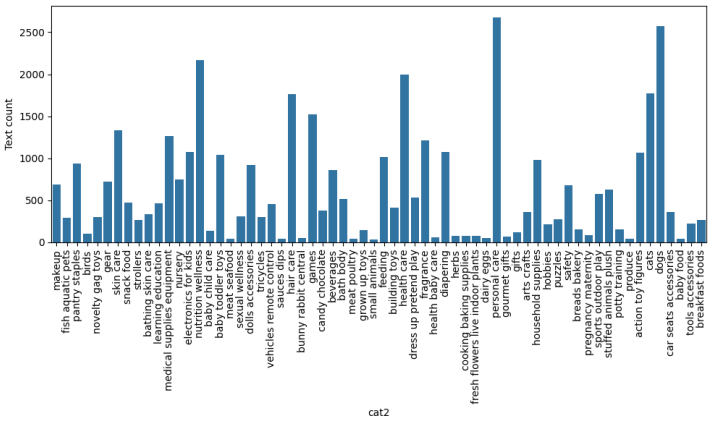
El dataset permite explorar distintas estrategias de clasificación jerárquica, entre ellas:

- **Clasificación plana:** Concatenación de los nombres de las clases (ej. "nivel1/nivel2/nivel3") y entrenamiento de un modelo multicategoría estándar.
- **Clasificación jerárquica por padre:** Uso de modelos en cascada donde primero se predice la clase de nivel 1, luego la de nivel 2 y finalmente la de nivel 3.
- **Enfoques avanzados:** Modelos secuencia-a-secuencia (*seq2seq*) en los que la entrada es la reseña y la salida es la secuencia de etiquetas jerárquicas.

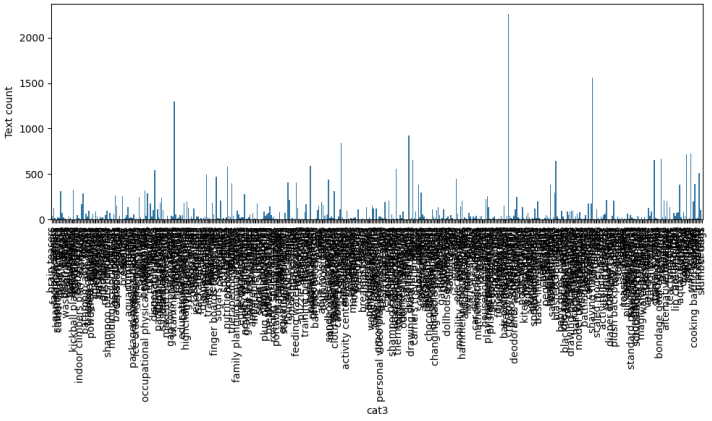
En este documento se detallarán los experimentos llevados a cabo para realizar clasificación por padre y clasificación por nivel.



(a) Distribución de etiquetas para la clase 1.



(b) Distribución de etiquetas para la clase 2.



(c) Distribución de etiquetas para la clase 3.

Figura 5: Distribuciones de las etiquetas en *dataset* de *Kaggle*.

4.2. Dataset sintético

También se ha producido un *dataset* sintético para la realización de pruebas iniciales. Este ha sido generado totalmente por *ChatGPT* y contiene tres niveles de jerarquía.

Este *dataset* contiene información sobre diversos temas científicos clasificados en tres niveles: dominio, subcampo y especialización. Cada fila proporciona un fragmento de texto explicativo sobre un concepto dentro de un área específica del conocimiento. Las columnas son:

- text: Explicación detallada de un concepto científico.
- domain: Campo general de estudio (Ej. Física, Química, Ciencias de la Tierra).
- subfield: Subcategoría dentro del dominio (Ej. Mecánica Clásica, Termodinámica, Química Física).

5. Implementaciones

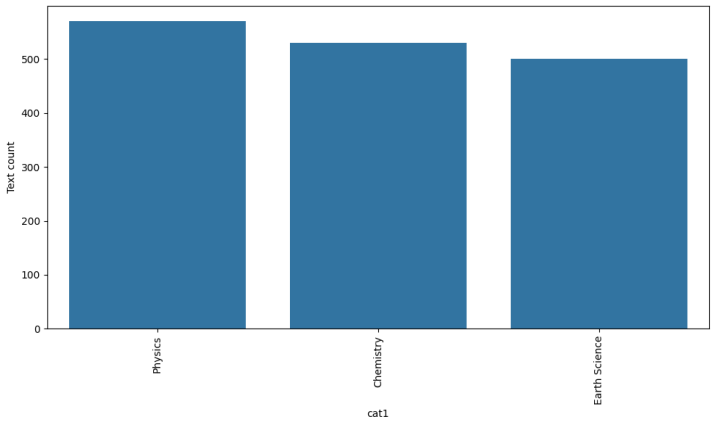
5.1. Naive Bayes

Existen múltiples enfoques para abordar el problema de *HTC*, y uno de los métodos más simples y eficientes es el clasificador **Naïve Bayes**, que se basa en la aplicación del Teorema de Bayes con la suposición de independencia condicional entre características.

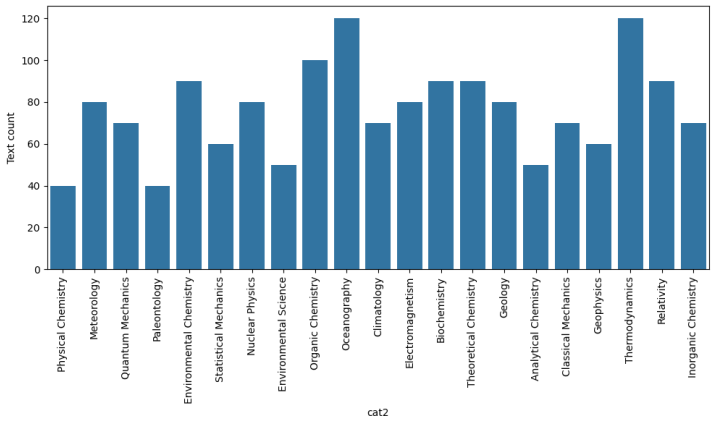
Este tipo de clasificador, junto a un pre-procesamiento adecuado, es un potente clasificador de texto para tareas simples. Por ello se propone como algoritmo base como ejemplo para demostrar la potencia de las arquitecturas propuestas.

5.2. Resultados de Naive Bayes

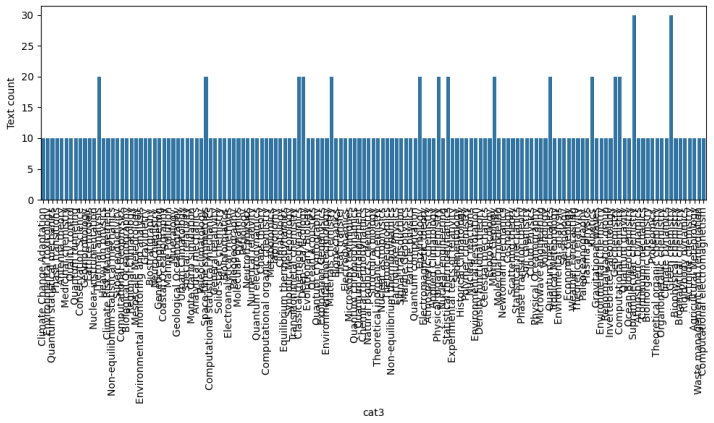
En la tabla 1 se pueden observar los resultados obtenidos por cada arquitectura. Obviamente los nodos raíz parten de la misma base en ambos y por ello obtienen los mismos resultados. La mejora empieza a observarse a partir del segundo nivel. La categoría dos mejora casi un 10 % en todas las métricas. Esto se debe a que el clasificador de la primera categoría es más fácil (hay menor número de clases), por lo que tiene mayor porcentaje de acierto. De esta manera, al filtrar las clases posibles para la segunda etiqueta con una primera predicción de la etiqueta padre, las clases predichas se reducen, los modelos de ese nivel son menos complejos y la clasificación mejora. En el nivel tres es donde de verdad se alcanza una mejora muy sustantiva.



(a) Distribución de etiquetas para la clase 1.



(b) Distribución de etiquetas para la clase 2.



(c) Distribución de etiquetas para la clase 3.

Figura 6: Distribuciones de las etiquetas en el *dataset* sintético.

Algoritmo	Métrica	cat1	cat2	cat3
NB per parent	Accuracy	0.6641	0.3510	0.2186
	F1	0.6354	0.3139	0.1715
	Recall	0.6641	0.3510	0.2186
	Precision	0.7758	0.4943	0.2636
NB per level	Accuracy	0.6641	0.2816	0.1010
	F1	0.6354	0.2232	0.0613
	Recall	0.6641	0.2816	0.1010
	Precision	0.7758	0.4955	0.2282

Cuadro 1: Resultados de Naive Bayes por categoría y métrica

5.3. LSTM

Las redes neuronales de memoria a corto y largo plazo (*Long Short-Term Memory*, *LSTM*) son un tipo de red neuronal recurrente (*Recurrent Neural Network*, RNN) diseñadas para manejar secuencias de datos. Su capacidad para capturar dependencias a largo plazo en los textos las hace especialmente adecuadas para tareas de clasificación de textos, incluyendo la clasificación jerárquica de textos (*HTC*).

A diferencia de los métodos tradicionales como *Naïve Bayes*, que tratan las palabras de manera independiente, *LSTM* mantiene un estado interno que le permite recordar información relevante en la secuencia, lo que ayuda a mejorar la clasificación al considerar el contexto del texto completo.

De igual forma que con *Naïve Bayes*, se realizarán modelos basados en arquitecturas *local per level* y *local per parent* y se compararán los resultados.

Además se aprovecha para implementar una red sencilla basada en *PyTorch* y aprender el *framework*.

5.4. Resultados de LSTM

En la tabla 2 se presentan los resultados de dos variantes del algoritmo LSTM: *LSTM-per-parent* y *LSTM-per-level*, evaluados en tres categorías distintas (cat1, cat2, cat3) a través de varias métricas: precisión (Accuracy), F1, recall y precisión (Precision). Ambos enfoques muestran resultados similares en la primera categoría (cat1), donde los valores de las métricas son prácticamente idénticos. Sin embargo, es en las siguientes categorías donde se observa una diferencia notable. En *LSTM-per-parent*, la segunda categoría (cat2) presenta un rendimiento significativamente superior en comparación con *LSTM-per-level*, especialmente en las métricas de F1 y precisión. Esto sugiere que al considerar la etiqueta padre, el modelo puede filtrar mejor las posibles clases, mejorando el rendimiento en

Algoritmo	Métrica	cat1	cat2	cat3
LSTM per parent	Accuracy	0.6950	0.4253	0.2969
	F1	0.6600	0.393	0.2526
	Recall	0.6950	0.4208	0.2940
	Precision	0.8000	0.5900	0.3270
LSTM per level	Accuracy	0.6950	0.3500	0.1842
	F1	0.6600	0.3800	0.1400
	Recall	0.6950	0.3591	0.1816
	Precision	0.8000	0.5702	0.2900

Cuadro 2: Resultados de LSTM por categoría y métrica

categorías más complejas. La tercera categoría (cat3), aunque con menores mejoras, sigue mostrando un patrón similar, con *LSTM-per-parent* superando a *LSTM-per-level*, lo que resalta la importancia del modelo en la predicción de categorías más difíciles.

6. Conclusión