UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

CLASIFICACIÓN DE PRODUCTOS A TRAVÉS DE ÁRBOLES DE CATEGORÍAS PARA E-COMMERCE

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO

MATÍAS IGNACIO GARCÍA GUTIÉRREZ

PROFESOR GUÍA:
JUAN BARRIOS NUÑEZ

MIEMBROS DE LA COMISIÓN:
JORGE SILVA SÁNCHEZ
JOSÉ SAAVEDRA RONDO

SANTIAGO DE CHILE
2020

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: MATÍAS IGNACIO GARCÍA GUTIÉRREZ
FECHA: ENERO 2020
PROF. GUÍA: JUAN BARRIOS NUÑEZ

CLASIFICACIÓN DE PRODUCTOS A TRAVÉS DE ÁRBOLES DE CATEGORÍAS PARA E-COMMERCE

El presente documento expone el trabajo de construcción y evaluación de clasificadores jerárquicos utilizando técnicas de *Machine Learning* con el objetivo de resolver el problema de clasificación de subcategorías en un e-commerce. Para esto se utilizan dos tipos de clasificadores jerárquicos: Uno local que utiliza un clasificador por nodo de decisión en el árbol y otro global donde se utiliza un solo clasificador para la predicción de las subcategorías.

El problema a resolver, nace de la necesidad de automatizar la elección de subcategorías que el usuario debe ingresar cuando ofrece sus productos a través de un marketplace, ya que éste puede producir potenciales errores de mala clasificación, reduciendo la efectividad del marketplace para vender ese producto.

La forma de resolver el problema planteado, consta de estudiar las técnicas actuales de clasificación jerárquica, procesamiento de lenguaje natural y visión computacional, además de formas de evaluar el problema de clasificación jerárquica. Para posteriormente, construir y entrenar los clasificadores jerárquicos de tipo global y local y ser utilizados como una automatización al problema de elección de subcategorías. El clasificador jerárquico global corresponde a una red neuronal inspirada en las técnicas aplicadas en Yolo 9000 a la que bautizo como HiGNet (Hierarchical Global Network) y el clasificador local se construye utilizando un clasificador RandomForest por nodo de decisión del árbol de categorías.

Con los resultados obtenidos, se concluye que el clasificador jerárquico local con descriptores de texto obtiene los mejores resultados en todas las métricas utilizadas asociadas a la precisión del clasificador en las primeras cuatro subcategorías, pero con la desventaja de tener intenso uso de recursos computacionales, mientras que HiGNet es liviano en uso de recursos computacionales, pero presenta una importante baja de precisión en las subcategorías más específicas.

Para Gloria y Manuel.

Agradecimientos

Agradezco a todos los involucrados de manera directa o indirecta con la realización de este trabajo.

Saludos, Matías García :).

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Motivación y antecedentes

Hoy en día han prosperado en todas partes del mundo los llamados **online marketplace**. Estos son sitios web de ventas de artículos, los cuales han causado una gran revolución a la hora de ofrecer productos por parte no solo de los grandes agentes del retail, sino también por parte de los usuarios a través de sitios dedicados al comercio entre ellos. Estos sitios permiten ofrecer productos por internet, ya sea de manera local, nacional o internacional a través de unos pocos clicks, llevando así a nuevos niveles de comodidad y eficiencia a la hora de comprar cualquier tipo de producto. Actualmente esta industria crece a tasas más altas comparada con la forma tradicional de comercio y desde el 2005 hasta el año 2018 ha triplicado su participación en el panorama total de ventas en retail [4]. Usualmente los productos ingresados a un marketplace agrupan su información en las llamadas **fichas** que corresponden en su formato más simple a un título, una descripción del producto, fotografías y una **subcategoría**. La subcategoría ayuda a la sugerencia de productos dentro de la misma plataforma y estas están indexadas en una estructura de tipo árbol estrictamente jerárquica a la que denominaremos **árboles de categorías**. Un ejemplo de una ficha corresponde a la figura 1.1.

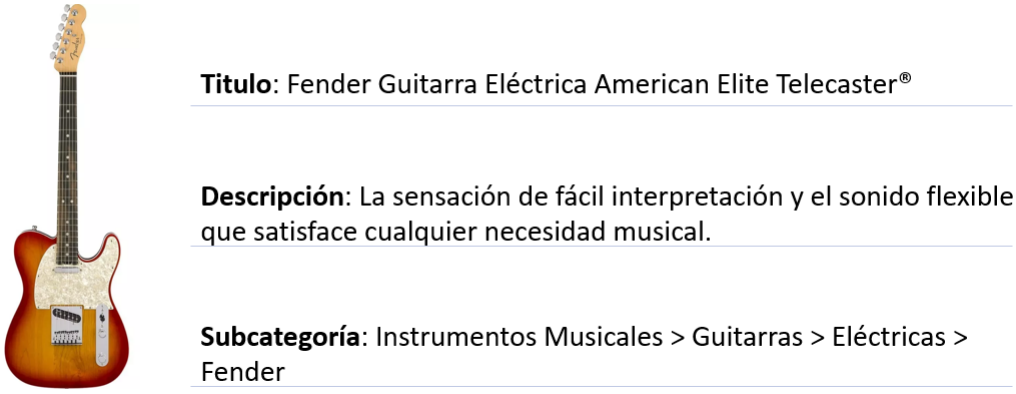


Figura 1.1: Ficha de guitarra eléctrica Fender Telecaste

En la actualidad lo más común es que los marketplace posean árboles de categorías donde los usuarios buscan los productos deseados, un ejemplo de esto puede ser encontrado en la figura 1.2, en donde se muestra las clases padres del árbol de categorías de tres marketplace. Dentro de estas plataformas existe una tendencia de poseer árboles con mayor cantidad de clases y especificad en sus hojas, llevan así a enormes árboles de categorías con gran número de clases sobre las cuales el usuario debe seleccionar el camino más adecuado dentro del árbol a la hora de ofrecer sus productos.

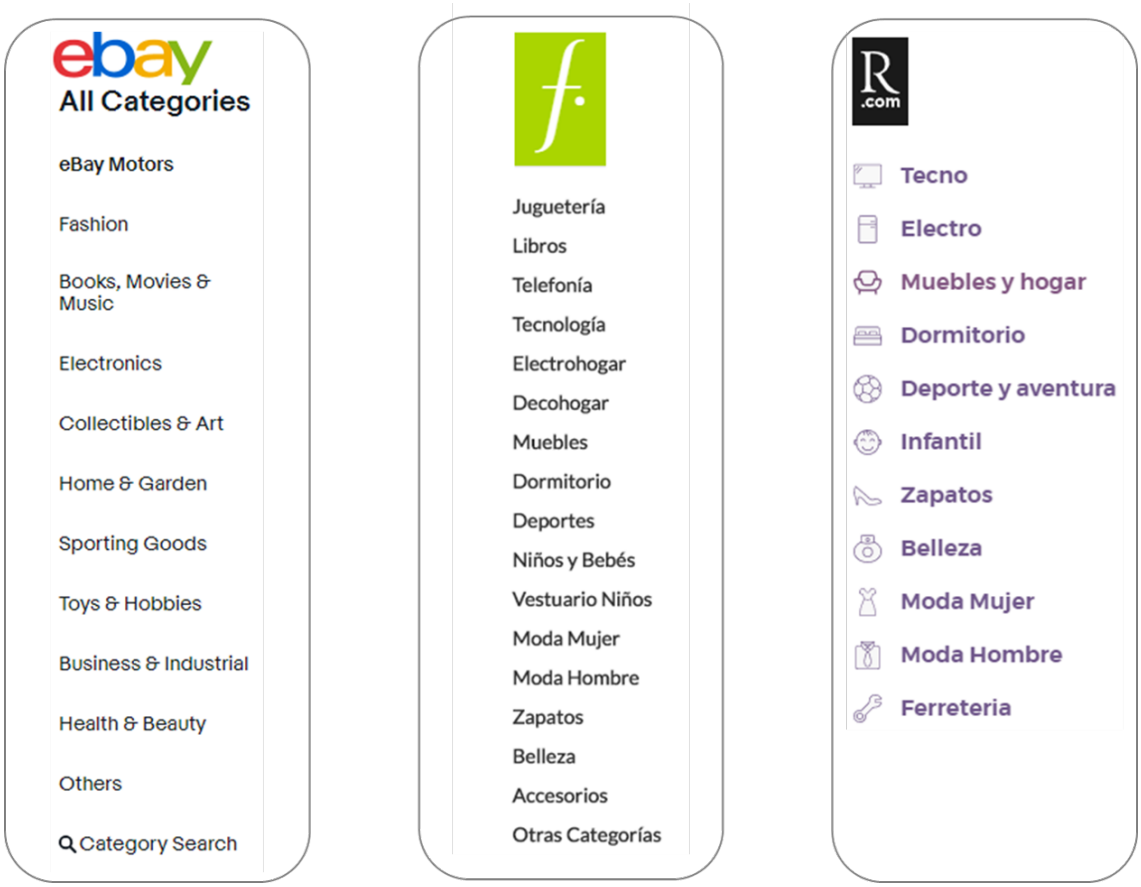


Figura 1.2: Ejemplos de categorías principales de tres e-commerce: Ebay, Falabella, Ripley respectivamente.

Este fenómeno puede ser beneficioso a la hora de generar una recomendación, debido a que permite ser más específico en la oferta de productos, sin embargo, puede ser perjudicial al momento que un usuario ofrezca un producto a través de la plataforma, ya que este puede carecer de la experticia necesaria para categorizar correctamente el objeto llevando así a un **etiquetado incorrecto**. Esta mala clasificación tiene efectos negativos en la recomendación del producto y eventualmente en la venta del mismo llegando a afectar económicamente tanto a la plataforma como al usuario. Un ejemplo de la forma en que actualmente se le pide la categoría a los usuarios puede ser vista en la figura 1.3, en donde se observa una lista de categorías sobre las cuales el usuario debe elegir y es su responsabilidad seleccionar el etiquetado correcto.

Hoy en día estas tiendas digitales deben asignar recursos y tiempo para limpiar y ordenar

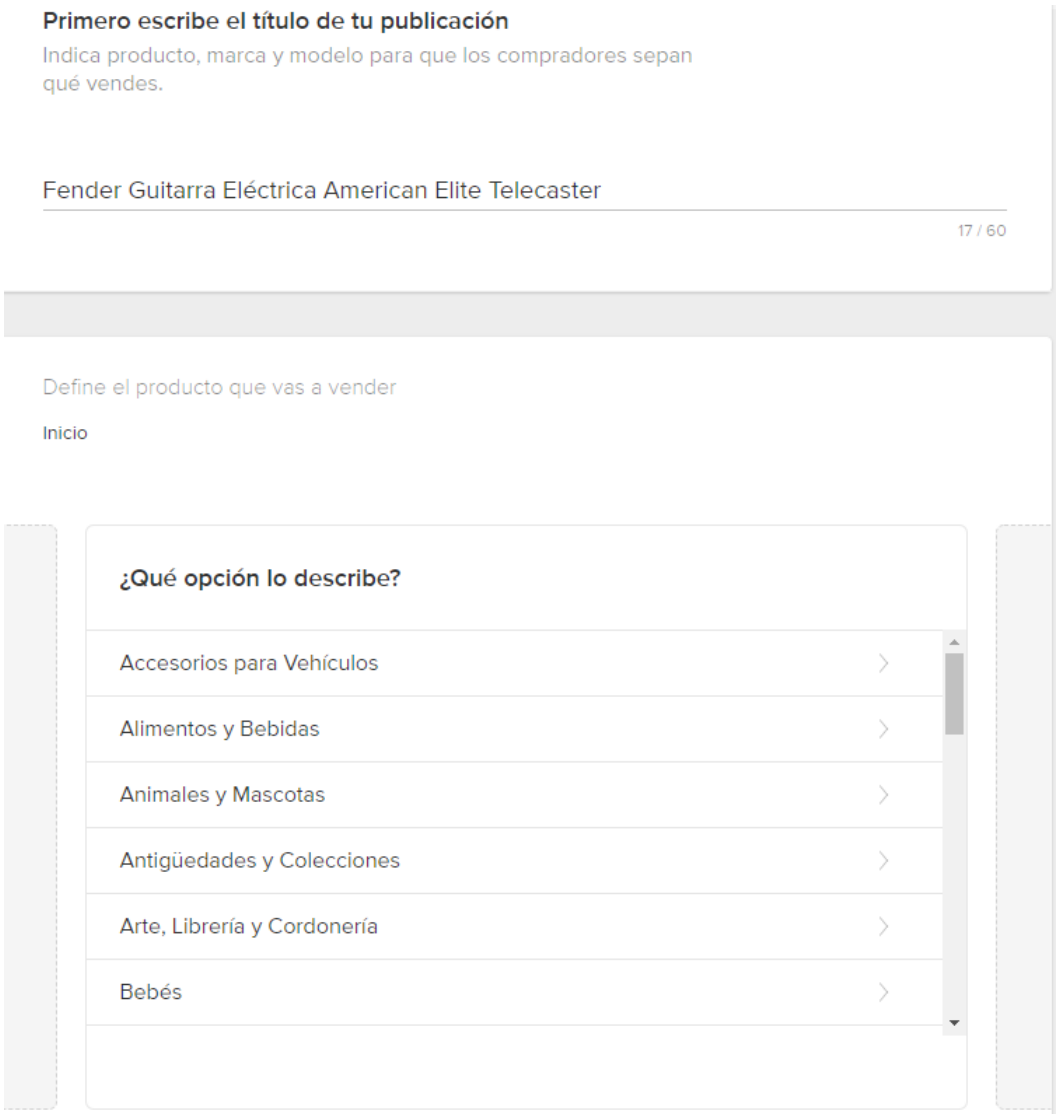


Figura 1.3: Ejemplo de registro de un producto en MercadoLibre

estos productos con el fin de disminuir el daño producido por la mala categorización. La práctica más común dentro de los e-commerce actuales es otorgar un servicio de recomendación de categoría a través de **predicción de texto**. Estos usan generalmente el título del artículo como fuente de información para inferir el camino más probable dentro del árbol de categorías. Si bien esta solución es un buen punto de partida, puede no ser suficiente.

Esto demuestra la necesidad de un clasificador multimedia que integre otras fuentes de información además del título del producto a clasificar. Una de las fuentes de información que puede ser utilizada para este propósito es una **fotografía del producto** esperando que ésta logre cubrir las deficiencias de la predicción por texto.

Debido a la importante llegada del comercio digital en la vida cotidiana de las personas durante los últimos años, se vuelve **crítico obtener una automatización fiable** para este problema. Una solución confiable permitiría a los usuarios utilizar las plataformas digitales sin complejos árboles de categorías, mientras que al mismo tiempo la plataforma pueda operar y hacer recomendaciones de forma interna de manera correcta, mejorando la experiencia por parte del usuario y a su vez, el desempeño de la plataforma.

Una alternativa para mejorar estos sistemas, como se mencionó anteriormente, es **reforzar** los sistemas de predicción actuales a través del uso de texto junto a las imágenes entregadas por el usuario o incluso, en algunos casos, utilizar sólo uno de éstos para lograr una predicción confiable. Dentro de lo que se espera de esta solución propuesta es la posibilidad de obtener una predicción del camino sugerido dentro del árbol de categorías y entregárselo al usuario, todo esto en tiempos que permitan hacer viable su uso en línea.

La propuesta de este trabajo de memoria es abarcar la clasificación jerárquica para solucionar el problema de automatización de categorización de productos dentro de una plataforma de e-commerce. Esto se realizará utilizando la información disponible en una ficha simple, es decir, **imagen** y **texto** con los dos enfoques más comunes dentro de la clasificación jerárquica: La **clasificación jerárquica local y global**.

1.2. Descripción del problema

El problema abordado corresponde a la automatización de manera fiable de la clasificación de subcategorías de nuevos productos en un marketplace, en donde el usuario ofrece sus productos. Esta solución ayuda a mejorar tanto la experiencia del vendedor como el funcionamiento de la plataforma. Actualmente esta tarea se realiza por clasificación de texto, en particular, utilizando el título de la ficha del producto. Sin embargo, el texto puede no ser suficiente, ya que existen casos donde palabras iguales en contexto diferentes pueden tener significados semánticos totalmente distintos, por ejemplo, consideremos el caso de batería como forma de almacenamiento eléctrico y batería como instrumento musical, un clasificador por texto podría confundir estas dos clases y llevar a un error en su predicción. Se cree que la forma de mejorar estos fallos es a través de la clasificación multimedia tomando no sólo la información entregada por el título del artículo, sino también la entregada por la imagen del producto.

La información para generar el dataset es extraída de un marketplace público con suficiente cantidad de datos para entrenar un modelo y deberán poseer una estructura de árbol jerárquica sobre la cual se trabajará. La información mínima de ficha con la que se trabaja es: Un título, una descripción del producto, una imagen y su jerarquía de subcategorías asociadas.

Uno de los grandes problemas de este trabajo es la gran variabilidad de imágenes dentro de las plataformas de comercio digital que permiten venta de productos entre usuarios, en donde existe un gran rango de calidad de imágenes, pasando por fotografías de tipo estudio con gran iluminación y contraste hasta imágenes obtenidas de manera casera por el mismo usuario con pobre iluminación y poca calidad. Además éstas vienen con gran variedad de tamaños y resoluciones, siendo necesario aplicar un pre-procesamiento adecuado para los algoritmos a utilizar.

En este trabajo se explorará y evaluará una solución utilizando información de texto, imagen y una combinación de ambos. Esta solución será implementada con dos tipos de clasificadores jerárquicos, uno local que corresponde a utilizar un clasificador por nodo padre dentro del árbol de categorías y uno global que corresponde a que un solo clasificador haga la predicción total de la jerarquía de subcategorías

1.3. Objetivos Generales

El objetivo general de esta memoria es la construcción y evaluación de un clasificador de tipo jerárquico de fichas de productos para dar solución al problema de predicción de subcategorías dentro de un árbol perteneciente a un e-commerce, esto utilizando como fuente de datos la imagen y/o el título de la ficha del producto.

1.4. Objetivos Específicos

- Construcción de una base de datos basada en el árbol de categorías de un e-commerce.
- Implementar un clasificador jerárquico local utilizando las características visuales extraídas de una red convolucional pre-entrenada en ImageNet y los vectores de texto extraídos utilizando FastText.
- Proponer y entrenar un clasificador jerárquico global de tipo red neural con una arquitectura inspirada en **Yolo 9000** utilizando características visuales extraídas de una red convolucional pre-entrenada en ImageNet y los vectores de texto extraídos utilizando FastText.
- Comparar y evaluar rendimiento de ambos sistemas implementados con vectores de características visuales, texto y mixto con reducción de dimensionalidad del vector visual por PCA.

1.5. Estructura de la memoria

Esta memoria presenta las siguientes secciones:

- **Capítulo 1**: Introduce el tema de memoria, importancia y objetivos a cumplir durante el trabajo.
- **Capítulo 2**: Corresponde a una revisión bibliográfica en donde se entregará el contexto sobre el área del aprendizaje de máquinas, redes neuronales, PCA como algoritmos de reducción de dimensionalidad, y algunas métricas aplicadas sobre los problemas de clasificación multi-clase y jerárquico.
- **Capítulo 3**: Muestra una revisión bibliográfica del estado del arte en clasificación jerárquica haciendo énfasis en la solución presentada en la red neuronal Yolo 9000.
- **Capítulo 4**: Presenta el diseño de la solución, explicando los procedimientos a seguir y sus razones.
- **Capítulo 5**: Exhibe el análisis y resultados de los experimentos propuestos.
- **Capítulo 6**: Expone las Conclusiones del trabajo realizado y plantea el trabajo futuro.

,

Capítulo 2

Marco Teórico

2.1. Aprendizaje de máquinas

Es una rama de la **inteligencia artificial (IA)** que busca en enseñar a las máquinas comportamiento inteligente, alejándose de su contraparte a la programación clásica de sistemas inteligentes, donde se buscaba utilizar un conjunto de reglas explícitas en la programación proporcionadas por un experto en el tema. El aprendizaje de máquinas busca obtener este conocimiento experto a través de los datos entregados, sin necesidad de mayor intervención humana.

En la literatura se habla de tres categorizaciones principales para este tipo de algoritmos:

Aprendizaje supervisado Consiste en la entrega de los datos de entrada con su salida deseada en su fase de aprendizaje, este conjunto de datos es generalmente conocido como el **conjunto de entrenamiento**. Al conocer las entradas y salidas de varios ejemplos, el algoritmo tratará de encontrar una función que lleve cada muestra del espacio de característica a su etiqueta correcta, además de generalizar a nuevas entradas.

Aprendizaje no supervisado De manera independiente el algoritmo debe ajustar los datos sin ninguna información de etiquetas. El ejemplo más común de este tipo de algoritmos es el de **clustering**, que consiste en la creación de agrupaciones de puntos similares de forma automática.

Aprendizaje reforzado Busca generar toma de acciones inteligentes por parte de un algoritmo en ambientes complejos. Para generar estos comportamientos se busca maximizar una **recompensa** dada hacía un problema. De esta forma el algoritmo no necesita de datos de entrenamientos, sino sólo del criterio de recompensa a maximizar en el ambiente.

2.2. Perceptrón

Una perceptrón o neurona artificial es una unidad de cálculo inspirada en la biología de una neurona real.

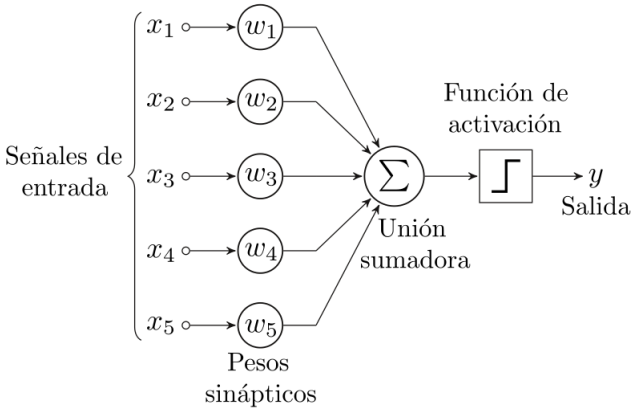


Figura 2.1: Modelo de neurona artificial estándar.

Donde:

- $(x_{\mathrm{i}}, ..., x_n)$: Vector de entradas del perceptrón. Esta parte es equivalente en una neurona biológica a las dendritas de otras neuronas.
- $(w_1, ..., w_n)$: Vector de pesos sinápticos. Equivalente a la conductividad que poseen las dendritas.
- $\sum$: Una función ponderadora es decir: $\sum_{\mathrm{i}=1}^{n} x_{\mathrm{i}} \cdot w_{\mathrm{i}}$
- Función de activación: Corresponde a una función no lineal para facilitar la aproximación de funciones más complejas que lineales.
- Bias (θ): Es un valor de referencia que usualmente se usa para desplazar la función de activación.

  Dejando como ecuación del perceptrón:

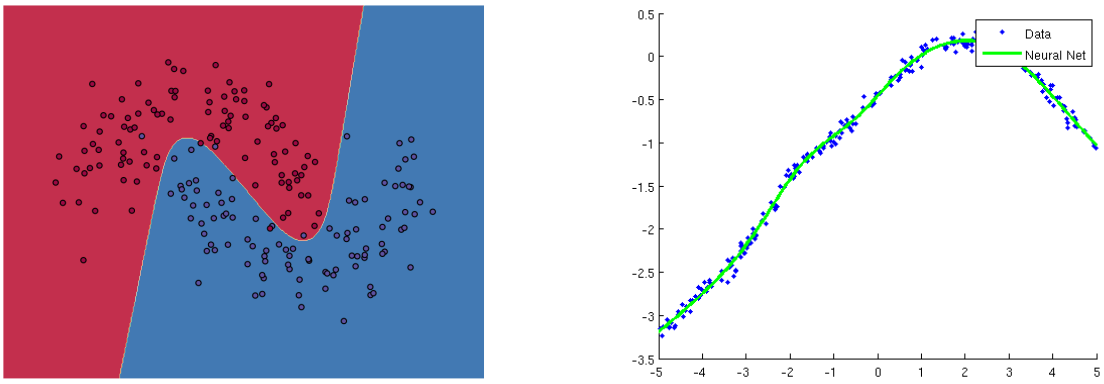
$$y = f(\sum_{\mathrm{i}=1}^{n} x_{\mathrm{i}} \cdot w_{\mathrm{i}} + \theta) \tag{2.1}$$

  Con $f(\cdot)$ como la función de activación.

2.3. Red neuronal feed-fordward

Las redes de tipo feed-fordward de una sola capa oculta han sido probadas como **aproximadores universales** de funciones [6]. Estas habilidades son principalmente explotadas en dos tipos de problemas: **clasificación** y **regresión**. El problema de clasificación corresponde a aproximar una función que lleve desde el espacio de características hasta el dominio de la etiquetas minimizando el error entre clasificación. Mientras que el problema de regresión

corresponde a ajustar una función determinada a partir de información conocida. Un ejemplo de estos dos casos se pueden ver en la Figura 2.2



(a) Clasificación

(b) Regresión

Figura 2.2: Casos comunes de uso de redes feed-fordward

Estas estructuras están compuestas por **capas** de las cuales podemos distinguir principalmente tres (Ver Figura 2.3).

- **Capa de entrada**: También denominada sensorial, está compuesta por neuronas que reciben las variables de entrada del problema.
- **Capa de salida**: Corresponde a la salida de la red neuronal.
- **Capa oculta**: Corresponde a la capa o conjunto de capas que se encuentran entre las capas de salida y entrada. Ésta no posee interacción directa con las variables de entrada ni salida y proporciona capacidad a la red de representar de mejor manera funciones complejas.

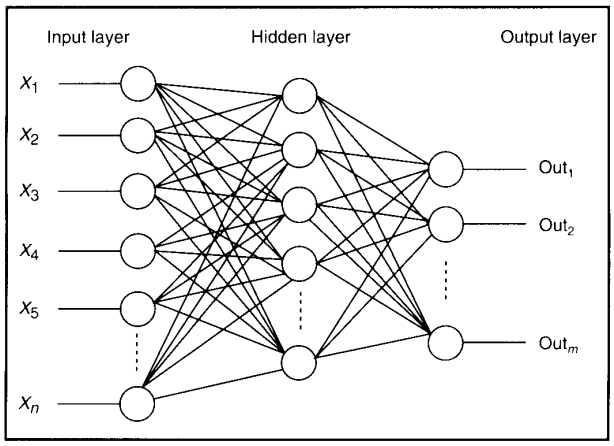


Figura 2.3: Esquema de red neuronal artificial

2.4. Redes neuronales convolucionales

Las redes neuronales convolucionales han sido el centro de atención en el campo de la visión computacional durante los últimos años. Aunque su uso se remonta a los años 90 para resolver problemas de detección de caracteres (Le Cun *et al.*, 1997), su popularización llegó el año 2012 cuando **Alex Krizhevsky** y su equipo ganaron la competencia **ILSVRC (ImageNet Large Scale Visual Recognition Competition)**[3] con la **AlexNet**[9], una red convolucional profunda diseñada para el reconocimiento de visual. Los dos factores más importantes para el florecimiento de esta tecnología se deben a la aparición de gran cantidad de datos generados por la internet y el incremento en poder computacional.

Hoy en día es común ver las tareas de reconocimiento visual del estado del arte resueltas por algún tipo de arquitectura de redes convolucionales.

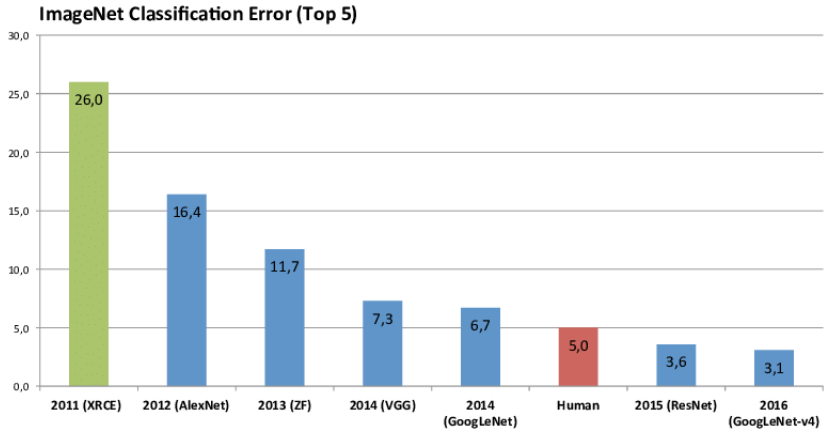


Figura 2.4: Ganadores ILSVRC (ImageNet Large Scale Visual Recognition Competition)

En la Figura 2.4 se puede ver el error top-5 alcanzado por los ganadores de la competencia ImageNet. Es posible notar que AlexNet venció al ganador del año anterior por casi un 10 % y desde el año 2015 este tipo de redes tienen desempeño sobrehumano.

Este tipo de redes en vez de utilizar la lógica clásica de las redes neuronales fully connected de tomar cada pixel como un input de la red, forma los denominados **kernel**, que corresponden a conjuntos de neuronas organizadas usualmente en una matriz cuadrada con la que luego se realiza la operación de convolución sobre los datos, permitiendo un ahorro de parámetros que la red debe aprender y a su vez actúa como regularización. Además ayuda a preservar una mayor información espacial, razón por la cual son populares en aplicaciones relacionadas al procesamiento de imágenes. Luego de finalizar el entrenamiento de este tipo de redes se obtienen una serie de filtros aprendidos por el algoritmo con el fin de identificar y generalizar sobre el tipo de objetos con el que fue entrenado. En la Figura 2.5 se muestra el proceso por el cual pasa una imagen por una red convolución, en donde se utilizan convoluciones y capas de pooling para crear **features maps** que corresponden a la información extraída por la red. Estos a medida que avanzan a través de la red y sus capas, se vuelven más específicos para la tarea en la que fueron entrenados. Finalmente esta arquitectura posee capas fully connected para llevar a cabo la clasificación.

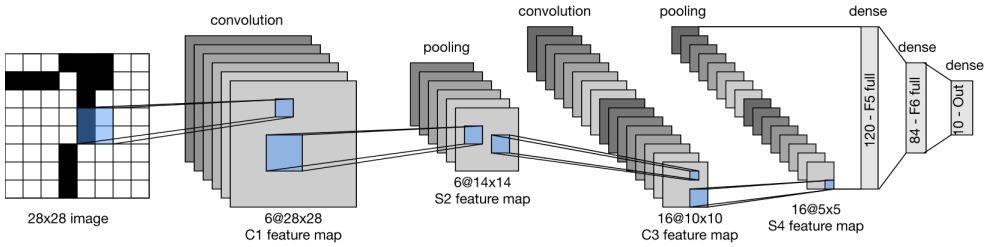


Figura 2.5: Flujo de datos en arquitectura LeNet-5. Las entradas corresponden a dígitos escritos manualmente y su salida es la probabilidad para cada una de las 10 opciones.

2.5. ResNet-18

La familia de redes tipo ResNet (*Residual Nets*) son desarrolladas por un equipo de visión perteneciente a *Microsoft Rearch* que con la ResNet-152, una red convolucional de **152 capas** obtuvieron el primer lugar en la tarea de clasificación de imágenes de la competencia ILSVRC 2015. Esta arquitectura busca dar solución al incremento en la dificultad de entrenamiento de las redes neuronales convolucionales a medida que se aumenta la cantidad de capas. Este es un concepto importante, ya que se ha visto en la práctica que mayor profundidad en las redes convolucionales va asociada a un incremento en su rendimiento [12]. En la Figura 2.6 se presenta el error top 3 de los ganadores del desafío de clasificación de imágenes en la competencia ILSVRC desde el año 2010 al 2015, además se apunta la cantidad de capas que posee cada uno de los algoritmos presentado. En este gráfico se aprecia una disminución del error a medida que aumentan las capas, como la tendencia a cada vez realizar redes más profundas.

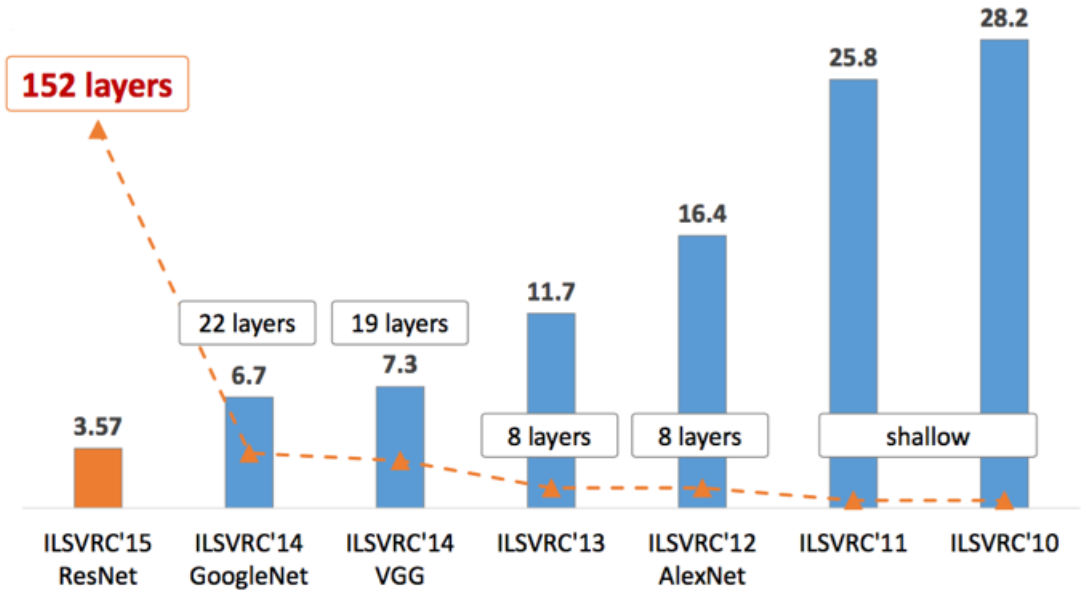


Figura 2.6: Error Top-3 de los ganadores de ILSVRC años [2010-2015]

Si bien ya se sabía que el aumento de profundidad de las redes generaba un incremento en su desempeño, en la Figura 2.6 se aprecia que aumentos considerables en profundidad no ocurrieron hasta la aparición de la ResNet, debido a que en la práctica se observa que al apilar más capas en una red su desempeño comienza a decaer, esto indica que el sistema se vuelve más difícil de optimizar [8].

Con el fin de resolver los problemas planteados anteriormente se propone la arquitectura de redes residuales, éstas están constituidas por **bloques residuales**. Supongamos que la función a ajustar es $H(x)$ y se define la función residual como $F(x) := H(x) - x$, que corresponde a la diferencia entre la función a ajustar y la entrada, es decir, un residuo. Como se ve en la Figura 2.7 la expresión de la función a ajustar $H(x)$ se puede expresar como $H(x) = F(x) + x$ y puede ser construida sumando la salida de la red con una conexión atajo desde la entrada. Esto funciona bajo el supuesto de que es más fácil ajustar la función residual que la original.

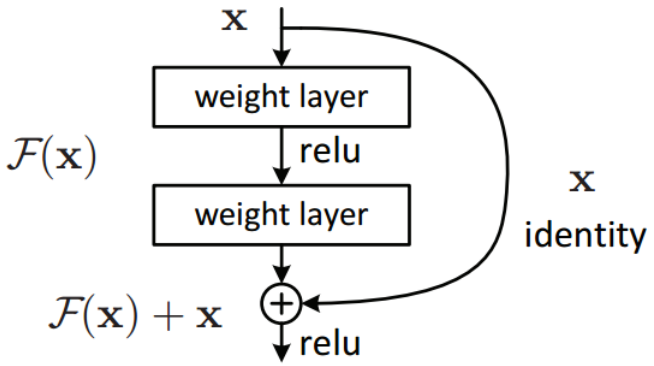


Figura 2.7: Bloque residual

2.6. Principal Component Analysis

Es un análisis matemático para la reducción de dimensionalidad de los datos. La filosofía de este algoritmo es hacer una proyección de los datos sobre los ejes de mayor variabilidad y a su vez disminuir el error de proyección. Esto se debe a que la variabilidad se puede entender como una medida de información de cuánto aportan los ejes a los datos. Cabe destacar que los nuevos ejes no necesariamente coinciden con los originales, sino que pueden ser una combinación lineal de ellos.

Varianza La varianza es la medida de dispersión de los datos con respecto a su promedio, en otras palabras mide qué tan alejados están los datos de su media.

Se puede expresar matemáticamente de la siguiente forma:

$$var(x) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N-1} \tag{2.2}$$

En donde:

- x_{i}: Muestra número i de la muestra.
- $\bar{x}$: Promedio de las muestras.
- N: Tamaño de la muestra.

Covarianza La covarianza mide cómo afectan los cambios en una variable a otra. Los elementos de esta matriz siempre son comparaciones entre dos dimensiones, si ésta es negativa, entonces indica que la segunda variable decrece cuando aumenta la primera, muentras que si es positiva aumentan juntas. Esta matriz es **simétrica** y en su diagonal posee la varianza de cada eje. Esto puede ser escrito de la siguiente forma:

$$cov(x,y) = \frac{\sum_{\mathrm{i}=1}^{N}(x_{\mathrm{i}} - \bar{x})(y_{\mathrm{i}} - \bar{y})}{N-1} \quad o \quad cov(X) = \frac{1}{N-1}\hat{X}\hat{X}^T \tag{2.3}$$

Donde $\hat{X} = X - \bar{X}$.

Vectores y valores propios Los vectores y valores propios son todos los que cumplan la siguiente ecuación:

$$Av = \lambda v$$

En donde A es una matriz cuadrada de nxn que representa la transformación lineal, v corresponde al vector propio y λ es el valor propio de ese vector.

El **eigenspace** es un espacio en donde cualquier vector que entre ahí se considera como un vector propio y no cambiará su dirección, pero difiere su magnitud al pasar por la transformación lineal A. Este conjunto puede ser entendido como $\{v \in V | Av = \lambda v\}$

Algoritmo Uno de los objetivos del PCA es obtener ejes, tales que exista **independencia lineal, minimizar el error de proyección y obtener ejes de mayor variabilidad**. Esto se puede expresar como una diagonalización de la matriz de covarianza, debido a que éstas serán cero.

$$S = \frac{1}{N-1}\hat{X}^T\hat{X} = PDP^T \tag{2.4}$$

En la ecuación 2.4, $\hat{X}$ son los datos centrados en su media y P es un la matriz de nxn compuesta de vectores propios ortogonales de $\hat{X}$ ordenados de la siguiente forma:

$$|\lambda 1| > |\lambda 2| > |\lambda 3| > ... > |\lambda n| \tag{2.5}$$
$$P = [v_1|v_2|v_2|...|v_n] \tag{2.6}$$

Por otro lado, D corresponde a los valores propios en su diagonal en el mismo orden anterior. Este orden es debido a que en esta transformación los **nuevos ejes son los vectores propios** y se ordenan de mayor a menor con respecto a su valor propio (asociado a la variabilidad).

La reducción de dimensionalidad se expresa de la siguiente forma:

$$\hat{Y} = P_k \hat{X} \tag{2.7}$$

Siendo $\hat{Y}$ los datos reducidos a una dimensión k, P_k la matriz de los k vectores propios con mayores valores propios y $\hat{X}$ los datos centrados originales.

2.7. Bag of Words

Bag of Words (BOW) es un modelo que busca la representación de texto arbitrario en un vector de largo fijo que se pobla usando la frecuencia de cada palabra dentro del mismo y que estas a su vez pertenezca a un vocabulario definido. A este proceso se le llama **vectorización**.

Tomemos el ejemplos de un conjunto de vocabulario que considere las siguientes palabras: **el**, **gato**, **sienta**, **sombrero** y **tiene**. En la Tabla 2.1 se puede ver la construcción de los vectores sobre ejemplos de texto considerando este vocabulario de palabras.

Documentos	el	gato	sienta	sombrero	tiene
el gato se sienta	1	1	1	0	0
el gato tiene un sombrero	1	1	0	1	1
el gato gato gato	1	3	0	0	0

Tabla 2.1: Ejemplo de BoW en tres documentos distintos.

Cabe destacar que esta técnica al solo contar palabras de los documentos pierde la información contextual asociada a la gramática del texto analizado.

2.7.1. N-grams

Bag of words al utilizar las palabras individualmente, no logra recoger la información asociada al orden de éstas en las oraciones. Una de las formas más usadas para incorporar

esta información al modelo es utilizando los llamados N-gramas de palabras. En la Figura 2.8 se puede ver un ejemplo de la formación de los tres primeros n-gramas sobre la frase *i love the food*, en donde n corresponde a la cantidad de palabras que se utiliza en un n-grama.

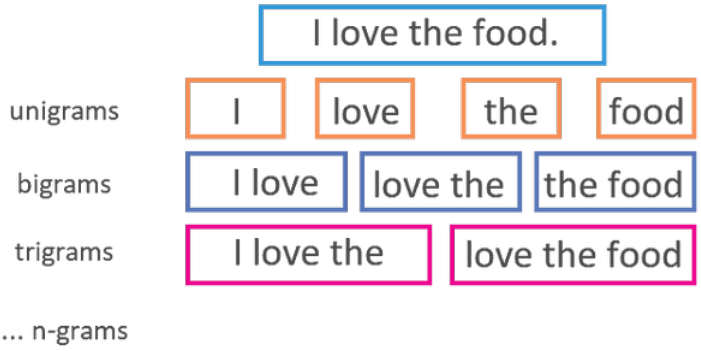


Figura 2.8: N-gramas formados para la frase *i love the food*

2.8. FastText

Este corresponde a un clasificador desarrollado por *Facebook AI* que promete ser **varios órdenes de magnitud** más eficientes en entrenamiento y evaluación que otros métodos de clasificación de texto y además, manteniendo resultados del estado del arte.

2.8.1. Arquitectura del modelo

Un buen modelo base para clasificación de texto es la representación de las oraciones como vectores utilizando **bag of words** (BoW) y luego entrenar un clasificador lineal sobre ellos. El problema de este acercamiento surge cuando el espacio de salida crece, es decir, en problemas donde el número de clases es muy alto. Una mejora para solucionar este obstáculo es la **factorización** de un clasificador lineal en una *low rank matrix* o utilizar un clasificador MLP. Esto permite encontrar separación de espacios no lineales para obtener una mayor generalización [1]. En la figura 2.9 se muestra la arquitectura del modelo *FastText*, en donde las entradas $x_1, ..., x_N$ corresponden a las característica de los N-gramas del texto de entrada. Estos vectores son embebidos y promediados en una capa oculta generando una variable única conocida como la **variable oculta** que resume la información del texto de entrada.

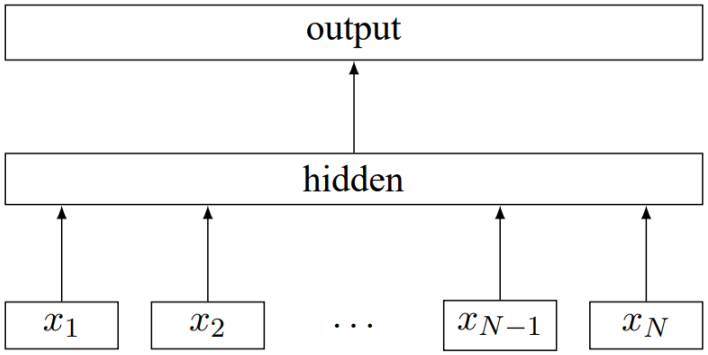


Figura 2.9: Diagrama de arquitectura del modelo Fasttext

Se utiliza la función softmax f en la salida del clasificador para estimar la distribución de probabilidad sobre las clases definidas. La función que busca optimizar este problema es la siguiente:

$$-\frac{1}{N}\sum_{n=1}^{N} y_n log(f(M(x_n)))$$

Donde x_n, es el vector embebido de la entrada n, y_n la etiqueta, M el modelo, f la función softmax y N el número de ejemplos.

2.9. Problemas de clasificación

El problema de clasificación consiste aproximar una función que mapea las variables de entrada en variables de salida discretas correspondientes a las clases o categorías. La función de mapeo busca a través de una observación predecir la clase o etiqueta correcta de ésta. En el contexto del aprendizaje una vez supervisado los datos son usualmente divididos en tres grupos: entrenamiento, test y validación. El conjunto de entrenamiento corresponde a los datos usados para enseñar al modelo a diferenciar y predecir la clase o etiqueta correcta de las observaciones, el conjunto de validación da una medida de la precisión en el entrenamiento que no dependa del mismo, ya que si solo se ve el desempeño de éste se podría caer en el sobreajustamiento y perder la capacidad de generalizar. Finalmente el conjunto de test a la evaluación final del modelo para obtener una estimación de su desempeño. Los tipos de clasificadores más comunes son los siguientes:

- **Binaria**: La entrada debe ser clasificada entre dos clases mutuamente excluyentes.
- **Multi-clase**: La entrada debe ser clasificada en sólo una de las n clases mutuamente excluyentes.
- **Multi-etiquetado**: La entrada es clasificada entre n clases distintas y su etiquetado es no excluyente, es decir un solo ejemplo puede ser etiquetado en más de una de las etiquetas fijas.

- **Jerárquica**: La entrada debe ser etiquetada de forma jerárquica, es decir, que se debe ir etiquetando en clases superiores (superclases), las cuales contienen otras clases inferiores (subclases).

2.9.1. Medidas de desempeño para clasificación

Algunos conceptos interesantes para verificar los resultados de una predicción sobre una clase son:

- **Verdaderos Positivos (VP)**: El sistema predice la clase correspondiente a la etiqueta.
- **Falsos Positivos (FP)**: El sistema predice la clase A, pero la observación corresponde a otra clase.
- **Verdaderos Negativos (VN)**: El sistema predice una clase diferente a la actual y corresponde a esa clase efectivamente.
- **Falso Negativo (FN)**: El sistema predice una clase diferente y su etiqueta verdadera es de la clase actual.

Precisión Corresponde a los verdaderos positivos predichos por el modelo sobre el total de consultas, esto se puede expresar matemáticamente como:

$$Precision = \frac{VP}{VP + FP} \tag{2.8}$$

Recall Corresponde a los verdaderos positivos predichos por el modelo sobre la cantidad total de positivos que el modelo debió encontrar. Esto puede ser expresado matemáticamente como:

$$Recall = \frac{VP}{VP + FN} \tag{2.9}$$

F1 Corresponde a una medida que intenta resumir la información entregada por las métricas precision y recall en una sola. Ésta puede ser expresada matemáticamente como:

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{2.10}$$

2.9.2. Medidas de desempeño clasificación jerárquica

Si se utilizan las métricas precision/recall de la clasificación multi-clase directamente en el problema de clasificación jerárquica se puede caer en una binarización de las respuesta, es

decir, si no se cumple el etiquetado exacto esto será tan erróneo como obtener un etiquetado parcialmente correcto, por ejemplo, es mucho peor confundir un perro de la raza *husky siberiano* con un microondas que confundirlo con un *pastor alemán*, por lo tanto, es necesario extender estas métricas para considerar resultados parcialmente correctos para el problema de clasificación jerárquica.

Como solución se introduce una traducción de estas medidas al mundo de la clasificación jerárquica, en donde ahora un camino parcialmente bueno también aporta a la métrica y esto puede ser expresado matemáticamente de la siguiente forma:

$$\begin{aligned} Hierarchical\ Precision &= \frac{\#CorrectSteps}{\#PredictedSteps} \\ Hierarchical\ Recall &= \frac{\#CorrectSteps}{\#TrueSteps} \end{aligned} \tag{2.11}$$

En donde:

- **CorrectSteps**: Corresponde a los pasos tomados correctamente por el clasificador dentro del verdadero camino (etiqueta).
- **PredictedSteps**: Cantidad de pasos que da la predicción dentro del árbol.
- **TrueSteps**: Corresponde a la cantidad de pasos tomados en el verdadero camino (etiqueta).

Este concepto es representado en un ejemplo en la Figura 2.10.

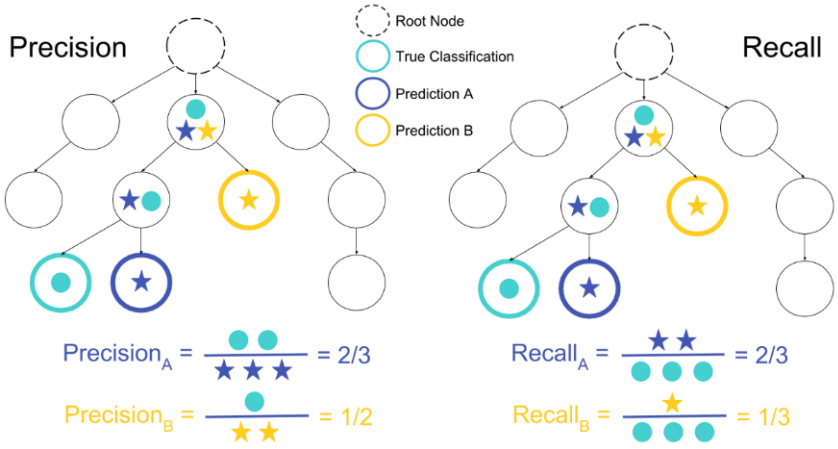


Figura 2.10: Métricas de clasificación jerárquica: En la izquierda se puede ver como la precisión jerárquica corresponde a la división entre la cantidad de pasos correctos en el total de pasos dados dentro de la jerarquía, mientras que en la derecha el recall es definido como el ratio entre los pasos correctos y la cantidad verdadera de pasos del ground truth.

Capítulo 3

Estado del Arte

En el contexto del problema de clasificación existe un nicho en donde las clases no son disjuntas, sino que poseen una estructura jerárquica y a esto se le da el nombre de *clasificación jerárquica* o por su nombre en inglés *hierarchical classification*. En [11] se define un problema jerárquico dada a una estructura de meta-clases que están compuestas en subclases formando típicamente estructuras de árbol. La manera típica de abordar estos problemas son a través de la optimización de funciones de costo local [11] o global [2].

Por optimización local se refiere a realizar optimización de un modelo por cada nodo de decisión del árbol jerárquico y por global se entiende optimizar un sólo modelo para toda la estructura.

Los métodos globales tienden a ser menos costosos computacionalmente, pero a su vez es probable que tengan dificultades a la hora de captar la información más especifica de las zonas más profundas del árbol, por otra parte, los métodos locales están compuestos por múltiples clasificadores y tienden a ser más costosos computacionalmente y lentos, además pueden caer en sobre ajuste de los datos.

3.1. Clasificación jerárquica: Yolo 9000

Es un sistema de detección de objetos, que promete tiempo real y precisión cercana al estado del arte. El problema de detección corresponde a poder encontrar en una imagen la localización (generalmente un bounding box) y la clase de una instancia de objeto. Un ejemplo de esto puede ser encontrado en la Figura 3.1.

Los datasets para detección de objetos son muy costosos de producir, debido principalmente a la dificultad de etiquetar una gran cantidad de imágenes con bounding box y clases. A causa de estas limitaciones Yolo 9000 incorpora la idea de utilizar datasets de detección para aprender el concepto de objectness (probabilidad de que sea un objeto) y localización de bounding box, mientras que utiliza datos de imágenes de clasificación para expandir su vocabulario.

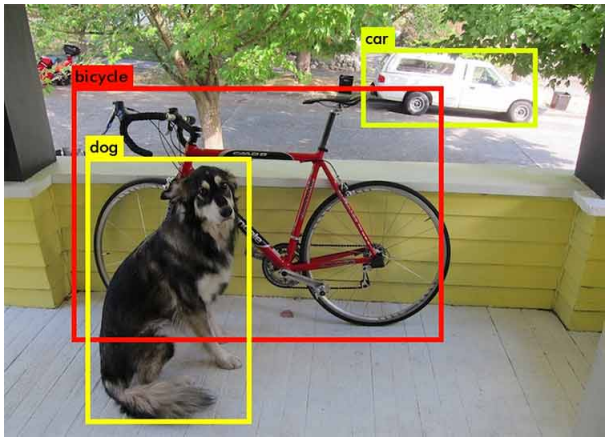


Figura 3.1: Detección en Yolo

Para probar este concepto se propone unir el dataset de detección **COCO** (Common Objects in Context) [13] que contiene 80 clases de objetos cotidianos e ImageNet que por otro lado, es un dataset de clasificación con más de 20.000 clases, que son extraídas de **WordNet** que corresponde a una base de datos que estructura los conceptos del lenguaje y forma de relacionarse. La práctica dominante en la clasificación de imágenes es utilizar etiquetas excluyentes, por ejemplo, un perro golden retriever y un labrador retriever pertenecen a la jerarquía de caninos retriever, pero en esta forma de clasificar son mutuamente excluyentes. Con el fin de evitar este problema Yolo 9000 presenta WordTree un árbol de conceptos visuales estrictamente jerárquico y se puede ver un ejemplo de este en la Figura 3.2.

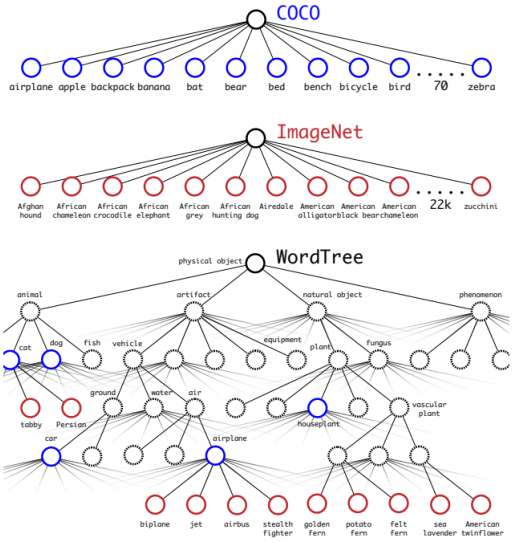


Figura 3.2: Unión de COCO e ImageNet para formar WordTree

La red entrenada en Yolo 9000 esta diseñada para predecir **probabilidades condicionales** en cada uno de los nodos de WordTree, por ejemplo, si se encuentra en el nodo Animal obtendrá probabilidades condicionales para sus nodos hijos, por ejemplo, perro, gato, pez, etc.

En caso de necesitar una probabilidad total sobre un nodo, simplemente se multiplican las probabilidades condicionales a través del camino del árbol. Por ejemplo, si se busca la probabilidad de que una foto sea un Norfolk Terrier, entonces calculamos:

$$\begin{aligned} Pr(\text{Norfolk terrier}) = Pr(&\text{Norfolk terrier}|\text{terrier}) \\ &*Pr(\text{terrier}|\text{hunting dog}) \\ &*\ldots* \\ &*Pr(\text{mammal}|Pr(\text{animal}) \\ &*Pr(\text{animal}|\text{physical object}) \end{aligned}$$

En donde se puede asumir $Pr(\textit{physical object}) = 1$ para el problema de clasificación (existencia de un objeto). Con este acercamiento se puede definir un límite de score de confianza en la multiplicación de las probabilidades condicionales, si se baja de este límite se detiene la clasificación en este nodo. Por ejemplo, podría identificar un animal de la clase perro si no tiene score de confianza suficiente para decir que es un Norfolk Terrier.

Para validar esta propuesta se utiliza la red Darknet-19 que corresponde a la red de clasificación que utiliza Yolo 9000 y se toman 1000 clases de ImageNet y se combinan con WordTree. Esto tiene como resultado un árbol con 1000 nodos de clases de ImageNet y 369 de nodos intermedios siendo en total 1369 nodos. Darknet-19 con este acercamiento reporta 71,9 % top-1 accuracy y 90,4 % top-5 accuracy.

Para realizar predicciones dentro de esta red se realizan softmax por partes al tensor de salida de dimensión 1396. Esto corresponde al equivalente a realizar esta operación sobre los nodos del árbol.

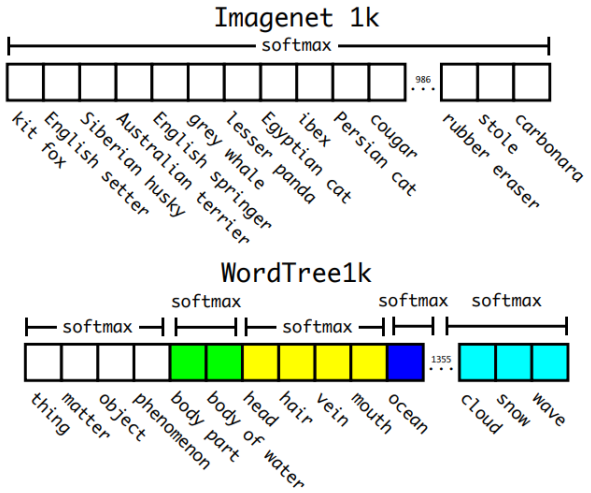


Figura 3.3: Mientras que en los problemas de clasificación comunes se utiliza un solo softmax para separar las clases en Yolo 9000 se realiza por nodo.

En la Figura 3.3 se puede ver como las clases están ordenadas en la forma que se recorre un árbol.

Capítulo 4

Diseño de la Solución

El objetivo final de este trabajo es la creación de un clasificador jerárquico para un árbol de categorías de un e-commerce que pueda ser usado como recomendador automático de categorías. Es decir, que sea confiable y que opere en tiempo real. Para lograr esto se seguirán los pasos expresados en la figura 4.1.

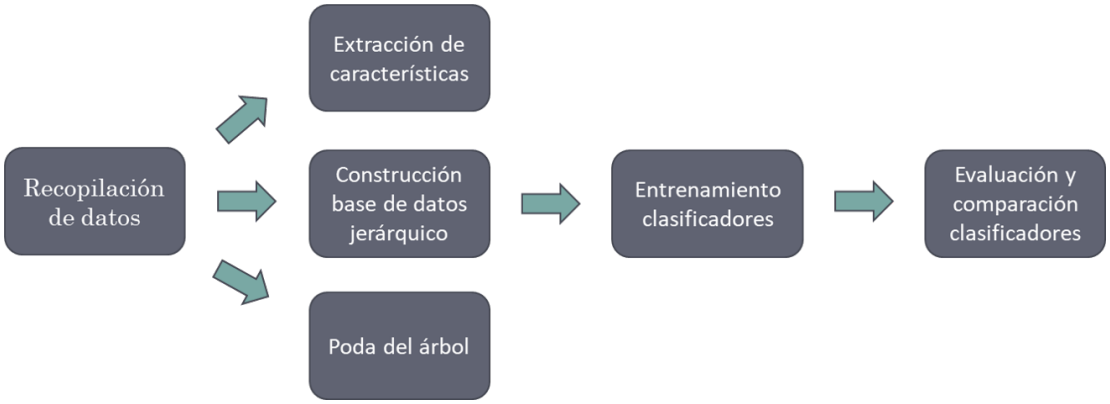


Figura 4.1: Diagrama de la metodología propuesta

En esta sección se explican los procesos que son llevados a cabo para la realización de este trabajo, en estos se explican los puntos principales como recopilación de datos, construcción de la base de datos jerárquica y finalmente, el entrenamiento y evaluación de dos tipos de clasificadores jerárquicos, más un clasificador caso base del problema.

4.1. Formalización del problema

El problema de clasificación jerárquica puede ser considerado como cualquier set de problemas que correspondan al conjunto (C, R), donde C es el conjunto de todas las clases en el problema y R es una relación de tipo "IS-A" que define un tipo de relación con las siguiente propiedades:

- **Asimétrica**: Todos los perros son animales, pero no todos los animales son perros.
- **Anti-reflexiva**: Ninguna clase puede ser hija o padre de sí misma.
- **Transitiva**: Todos los perros son mamíferos, y todos los mamíferos son animales, por lo tanto todos los perros son animales.

Las clases están estructuradas de acuerdo a una jerarquía que puede ser representada en la forma de un grafo dirigido, donde los nodos corresponden a clases y los arcos a la relación entre ellos [11]. La forma del grafo para este problema de clasificación jerárquica corresponde a la de un árbol, debido a la estructura heredada del árbol de categorías del e-commerce.

En este trabajo se abordarán los dos casos más comunes de la clasificación jerárquica correspondiente a un trabajo de clasificación jerárquica local inspirado en la arquitectura Local Classifier per Parent Node (LCPN) y otro clasificador jerárquico global inspirado en la ideas expresadas en Yolo 9000. Estos serán entrenados utilizando vectores de información visual obtenida de una red convolucional y de texto obtenidos utilizando FastText [1]. Estos se someterán a evaluación tanto de su rendimiento como sus tiempos de entrenamiento y ejecución pensando en un problema real.

4.2. Hardware

La máquina utilizada para este trabajo posee las siguientes características:

- **CPU**: Intel(R) Core(TM) i7-4770K CPU @ 3.50GHz
- **GPU**: Nvidia GeForce GTX 980 Ti
- **RAM**: 32 GB

4.3. Software

Para desarrollar este trabajo se ocuparon los siguientes software:

Python 3.7

- Librería scikit-learn 0.20.1
- Librería Pytorch 1.3.0
- Librería numpy 1.17.2
- FastText 0.9.1

4.4. Recopilación de datos

Con el fin de buscar una solución al problema descrito anteriormente es necesario obtener una base de datos de un **e-commerce real con una estructura jerárquica de clases**. Con esto en mente se decide utilizar como fuente de datos a la plataforma **Mercado Libre**, ya que cumple con las características de ficha, un árbol de categorías agrupado de forma estrictamente jerárquica y es uno de los e-commerce con mayor presencia en Chile y en Latinoamérica. MercadoLibre posee una API pública [10] disponible para los desarrolladores de todo el mundo, que puede ser usada para obtener las fichas de todos los productos publicados en un país y a su vez su árbol de categorías. Para este trabajo la información de interés que es extraída de las fichas de productos es la siguiente:

- **Título del producto**: Es la principal fuente de información de texto y contiene la información más fundamental acerca de un objeto.
- **Fotografía del producto**: Es la fuente de información visual que se usará para reconocer las categorías de los productos. En este caso se decide utilizar la imagen principal del producto bajo el supuesto de que es la más representativa y en su menor resolución para ahorrar recursos de almacenamiento y computacionales a la hora de entrenar modelos.
- **ID de la clase**: Es el código de identificación de la clase, se conforma de dos partes, donde la primera corresponde a letras que indican el país de la categorías y números correspondientes a la identificación del nodo dentro de ese país, por ejemplo, MLC1182 es un ID con el identificador "MLC" correspondiente MercadoLibre Chile y su clase es "Instrumentos Musicales".
- **ID de su clase padre**: Al igual que el ID propio corresponde a la identificación de la clase padre del nodo actual. Esto funciona como una referencia acerca de la posición actual en el árbol y, por lo tanto, permite recorrerlo.

4.5. Revisión y exploración de datos

Al ser la base de datos creada a través de la descarga de un catálogo de un e-commerce real, puede contener errores de mal etiquetado, por lo tanto se debe evaluar la calidad de éstos antes de realizar los experimentos, ya que los algoritmos de tipo supervisado son altamente susceptibles a datos ruidosos.

La revisión de los datos se centra principalmente en las imágenes, ya que son la fuente de datos sobre la que la plataforma tiene menos control. Como medida de ruido sobre la base de datos se busca estimar cuántas fotografías efectivamente corresponden a la jerarquía entregada por las fichas obtenidas de la plataforma, para esto se asume la simplificación de **solo utilizar las clases padres** sobre las cuales se revisan fichas de manera manual y se calcula el coeficiente I_{clase} (Ecuación 4.1) donde clase corresponde a una de las clases padres. Este coeficiente entrega una idea de que tan coincidente son las etiquetas otorgadas con la plataforma versus la clase que debería ser según lo observado por un humano.

$$I_{clase} = \frac{EtiquetasCorrectas}{RevisionesTotales} \tag{4.1}$$

Este indicador I_{clase} puede ser utilizado como medida para eliminar clases que posean demasiado etiquetado ruidoso, ya que esto podría disminuir el desempeño de los clasificadores.

En la Figura 4.2 se pueden observar ejemplos reales de revisión manual sobre la clase "Alimentos y Bebidas" donde un ejemplo corresponde a la clase y el otro no.

Imagen	Etiqueta	Revisión
	Alimentos y Bebidas	
	Alimentos y Bebidas	

Figura 4.2: Ejemplo de revisión manual de fotografías

Otros factores que se deben tener en cuenta sobre el árbol de categorías son: Distribución de datos de clases principales, cantidad de clases de primer nivel, distribución de clases principales, cantidad de niveles, cantidad total de clases nodo y cantidad de clases nodo hoja.

4.6. Extracción de características

Los algoritmos no pueden utilizar la información de las fichas de forma directa, por lo que se debe transformar a un espacio de vectores que los algoritmos comprendan. Para esto se procede a convertir tanto los títulos de los productos como sus imágenes en **vectores de característica**, a este proceso lo llamaremos **extracción de características**.

Extracción de características de imagen Una práctica común dentro del área de la visión computacional es la utilización de **redes neuronales convolucionales** como extractores de características para sus aplicaciones y esta es utilizada en variedad de sistemas del estado del arte. Estas redes neuronales son algoritmos basados en datos, es decir, estos obtienen el conocimiento para realizar una tarea asignada de un conjunto de entrenamiento de datos e intentan **generalizar** este proceso para datos fuera del conjunto de entrenamiento. Si bien este tipo de algoritmos son los que obtienen mejores resultados en la práctica, requieren

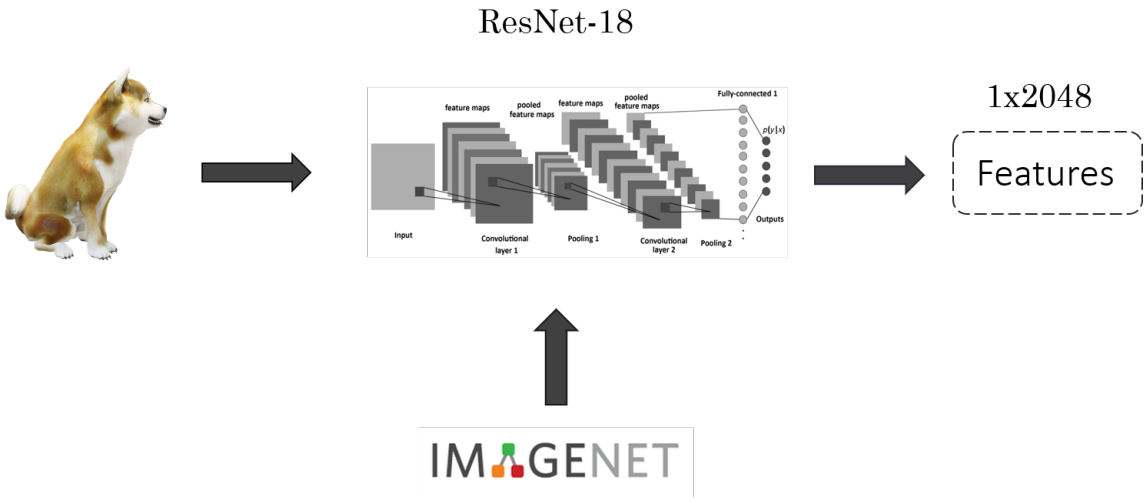


Figura 4.3: Esquema de extracción de características visuales para una ResNet-18 pre-entrenada en ImageNet.

una masiva cantidad de datos para su entrenamiento, por lo que una estrategia común es utilizar una red previamente entrenada en otro conjunto de datos masivo como extractor de características, bajo el supuesto de que esta red ya aprendió conceptos básicos de la visión como podrían ser detección de bordes y geometrías comunes. Este concepto es conocido como **transfer learning**.

En este trabajo se utiliza como extractor de características una red ResNet-18 pre-entrenada en la base de datos ImageNet. La forma más utilizada para ocupar estas redes como extractores de características es usar la salida de la capa anterior a la de clasificación como vector de características, esto debido a que el espacio de características generado por esta capa ha mostrado en la práctica buenos resultados para esta tarea de clasificación. Para el caso de la ResNet-18 el tamaño de sus vectores de salida es $1x2048$. Este proceso es ilustrado en la Figura 4.3 en donde se convierte la imagen de un perro *Akita* a través de una red pre-entrada a su respectivo vector de características respectivo.

Selección de características de imagen Debido al gran tamaño del conjunto de datos y del vector de características para imágenes se decide reducir la dimensionalidad del mismo, esto busca que tanto los tiempos de entrenamiento como de evaluación se reduzcan intentando evitar el deterioro del desempeño. Para realizar esta tarea se utiliza el algoritmo PCA para seleccionar y transformar las características con que se conserve mejor la información asociada a los vectores. Los pasos que se toman para realizar esta transformación son los siguientes:

- **Estandarizar datos**: Estandarizar los datos (promedio = 0 y varianza =1) es un proceso fundamental en la mayoría de las técnicas del aprendizaje de máquinas.
- **Selección de características**: Se usa PCA para seleccionar y transformar el espacio de característica, tal que se conserve el **95 %** de la varianza.

Extracción de características de texto Se utiliza **normalización** de texto como pre-procesamiento previo a los algoritmos. En este trabajo se utilizan las letras en su forma minúscula y la eliminación de caracteres especiales a través de la transformación del texto a código *ascii*. En la Figura 4.4 se puede ver la normalización sobre un título de un producto de la plataforma.

Figura 4.4: Normalización del título de un producto

La extracción de características se realiza entrenando un modelo de clasificación de texto de tipo no-supervisado en FastText sobre los datos del conjunto de entrenamiento y luego éste es utilizado para convertir los títulos de los productos al espacio de vectores. El modelo está basado en el trabajo de [?]. En la Figura 4.5 muestra el proceso de conversión de títulos a vectores de características.

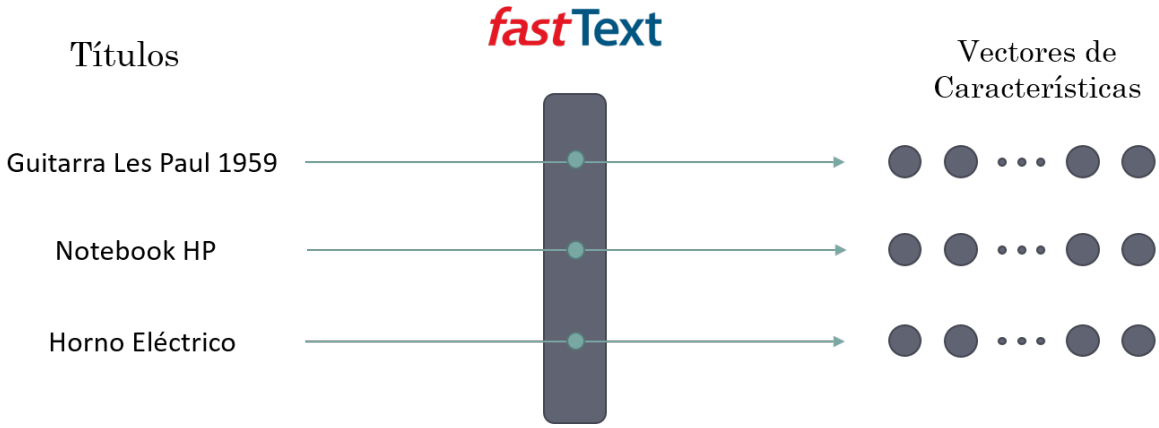


Figura 4.5: Esquema de extracción de características de texto

4.7. Construcción de base de datos jerárquica

Para el desarrollo de este problema es fundamental la definición de la jerarquía en la que estarán distribuidos los datos. Para esto se obtiene el árbol de categorías del catálogo con el que se trabajará y se guarda de manera local para su uso en los distintos procesos que se realizarán más adelante. Para esto **se organizan los datos en sus clases hoja**, es decir, cada par de datos (título, dirección de imagen de referencia) se guarda en conjunto con sus compañeros de categoría bajo la identificación de ID de su respectiva hoja. De esta forma los datos pueden ser recuperados y etiquetados de manera jerárquica cuando sea necesario, en particular, en este trabajo se utiliza la **búsqueda por profundidad** (DFS) para recuperar

y etiquetar los datos. En la Figura 4.6 se muestra el proceso de recuperación de datos de manera jerárquica subiendo las fichas desde las hojas hasta la clase principal.

Las ventajas de esta forma de almacenar los datos son las siguientes:

- Permite etiquetado jerárquico y recuperación de datos desde cualquier nodo perteneciente al árbol.
- Permite que el etiquetado sea dinámico y dependiente de la estructura actual del árbol, es decir, facilita el re-etiquetado de los datos en caso de modificaciones sobre el árbol de categorías.
- Facilita el entrenamiento de clasificadores locales por clase Padre (LCPN), ya que los datos suben a través del árbol desde las hojas a los nodos principales, permitiendo entrenar los nodos superiores a medida que van recibiendo los datos de nodos inferiores. Esto permite una forma jerárquica, eficiente y flexible de entrenar y evaluar clasificadores locales.

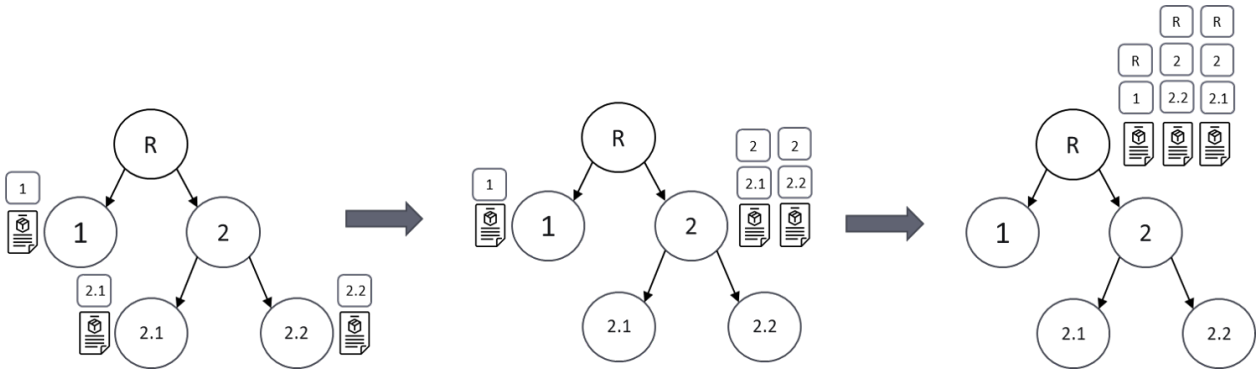


Figura 4.6: Diagrama de ejemplo de recuperación de datos de manera jerárquica. Los datos parten almacenados en las hojas 1, 2.1, 2.2 y estas luego suben por el árbol mientras se les agrega la etiqueta por cada nodo que pasen hasta llegar a la raíz.

4.8. Poda del árbol

Debido al gran tamaño y variedad existente de nodos dentro del árbol de categorías se decide realizar una poda de las categorías con los siguientes criterios:

- **Eliminación clase Otros**: Una práctica usual es crear una clase comodín donde van los artículos no definidos en las clases ya propuestas por la plataforma. Como ésta es muy amplia y contiene variedad de productos muy distintos, podría influir negativamente al desempeño del clasificador sin aportar mucha información útil, por lo tanto, se decide podar los nodos/clases que incluyan en su nombre las palabras Otros u Otras.
- **Clases con etiquetado ruidoso**: Al ser una base de datos extraída de un sitio marketplace real, se espera que contenga errores en la clasificación de los productos. Como criterio de limpieza se utilizará el coeficiente I_{clase} sobre cada una de las clases padres y se define un umbral donde se conservan todos los nodos sobre éste y el resto es eliminado, incluyendo todos los nodos correspondientes a su jerarquía hija. En la Figura

4.7 se ve un ejemplo de nodos de la clase principal con su identificación y su coeficiente I_{clase} el cual es usado para seleccionar los nodos que serán utilizados. Para este ejemplo se utiliza como umbral de limpieza 0,8.

Figura 4.7: Ejemplo de selección de nodos padres basados en su I_{clase}.

- **Suficientes ejemplos de entrenamiento**: Se debe fijar una cantidad mínima de ejemplos de entrenamiento para todos los nodos sobre los cuales se quiera aplicar el clasificador. Para esto se marca recursivamente empezando desde las hojas hasta la raíz los nodos que no posean suficientes datos como inválidos para el experimentos, un ejemplo de esto puede ser visto en la Figura 4.8, donde se muestran con una marca verde los nodos marcados con suficientes datos y con rojo los que no, todo considerando un mínimo de 300 ejemplos. Cabe destacar que el número de ejemplos de una clase padre es la suma de los ejemplos de sus clases hijas y por esta razón el árbol es recorrido de hojas a raíz.

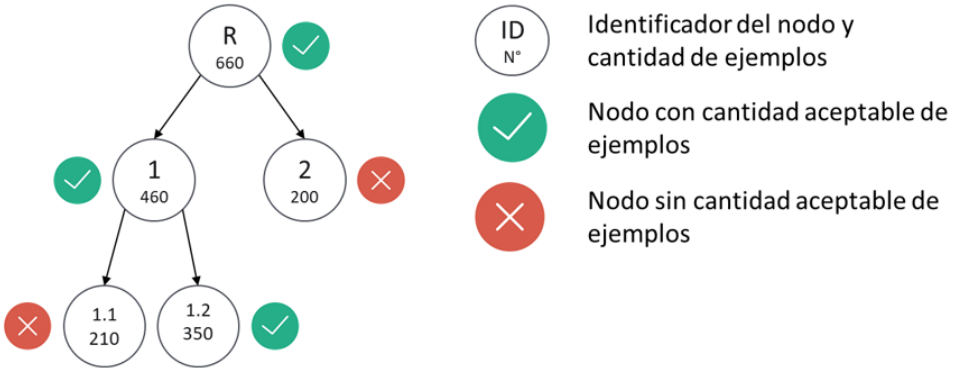


Figura 4.8: Ejemplo de selección de nodos basados en su cantidad de datos.

4.9. Clasificación jerárquica

Se busca evaluar las dos formas más comunes de clasificación jerárquica para resolver el problema propuesto, estos corresponden a un clasificador local y uno global. Todos los clasificadores son entrenados y evaluados con los siguientes tipos de descriptores: Imágenes, texto y una concatenación de ambos descriptores como forma de integración de información. Esta evaluación y comparación también se hará contra un caso base que corresponde a un clasificador que no integre información sobre el árbol de categorías sobre su estructura.

Caso base La forma más básica de la clasificación jerárquica puede ser entendida como un clasificador multi-etiqueta sobre todos los nodos del árbol esperando que el modelo prediga los nodos asociados al camino desde la raíz hasta la hoja real. Para la implementación de este clasificador primero se debe transformar el árbol a su forma plana. Para esto se recorre el árbol por nivel (Level Order Traversal) concatenando cada nivel con el siguiente, esto inspirado en los métodos implementados en [7]. En la Figura 4.9 se muestra la transformación de un árbol jerárquico a su forma plana, además se muestra un ejemplo de un camino desde su raíz a la hoja 1.1 marcado con estrellas y su equivalente en su forma plana y su etiqueta correspondiente.

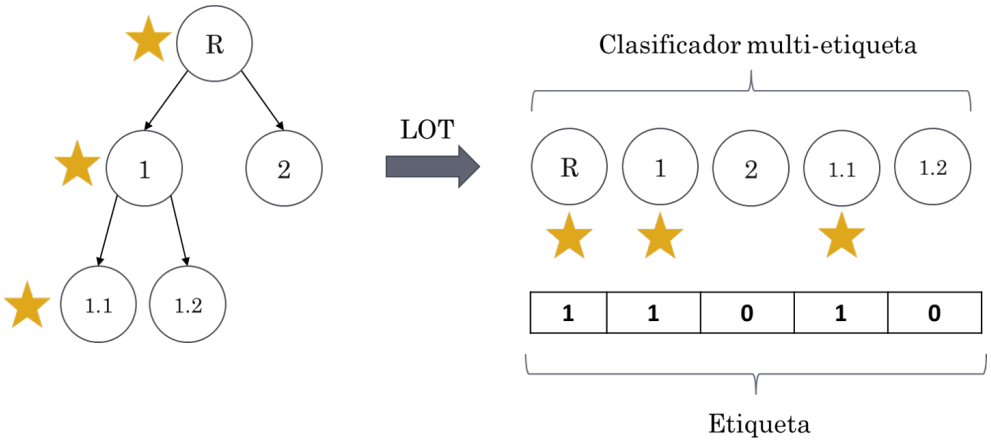


Figura 4.9: Ejemplo de transformación de árbol a forma plana

Clasificador jerárquico local Este clasificador jerárquico consiste en un conjunto de clasificadores estructurados de forma jerárquica, en particular, en este trabajo se utiliza **un clasificador multi-clase en cada nodo de decisión del árbol**. Cada uno de estos clasificadores debe discernir la clase del ejemplo sobre sus nodos hijos. En la Figura 4.10 se puede observa un ejemplo, en donde un árbol se divide en clasificadores locales por nodo de decisión para ser entrenados de manera independiente y luego estos se unen para formar el clasificador final.

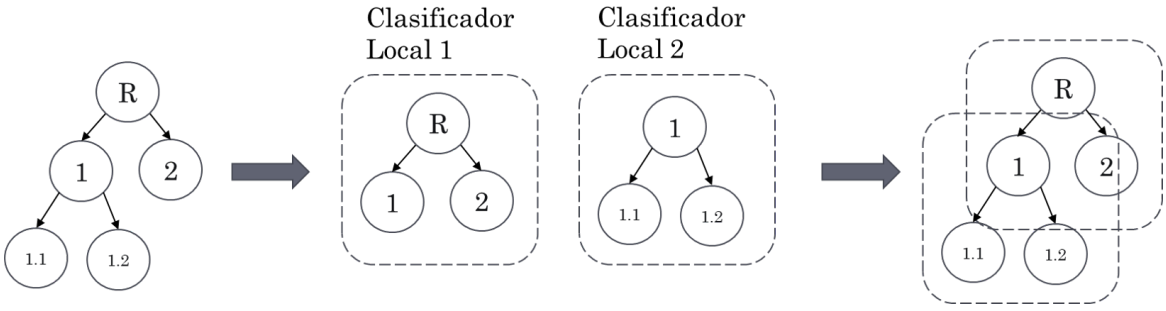


Figura 4.10: Ejemplo de construcción de clasificadores locales.

Debido al gran número de clasificadores a entrenar se debe elegir un modelo que posea baja cantidad de hiperparámetros para evitar complicados procesos de optimización de los

mismos. Con esto en mente se decide utilizar el modelo **Random Forest**, ya que este es posible de usar con un solo hiperparámetro: **el número de árboles de decisión** en el modelo enjambre.

El entrenamiento se realiza de manera jerárquica como fue señalado en la sección 4.6, en donde, se entrenan los nodos a medida que los datos suben desde las hojas hasta la raíz.

Clasificador jerárquico global Este clasificador jerárquico se inspira en el trabajo realizado en [7], en donde se utiliza una red neuronal que al igual que un clasificador multi-etiqueta realiza predicciones sobre todos los nodos del árbol en un clasificador plano, pero introduce la idea de calcular la función *Softmax* sobre todas las clases que posean el mismo padre. De esta forma se introduce la idea de distribuciones de probabilidad sobre las clases hijas de los nodos de decisión, siendo el de mayor probabilidad el nodo que se elige como predicción sobre ese nodo de decisión. La arquitectura propuesta es bautizada como la HiGNet (Hierarchical Global Network) una red neuronal jerárquica global. Esta estructura es una red neuronal fully connected con capa de entrada de tamaño igual a la característica a utilizar (texto, imagen o concatenación de ambos) y salida correspondiente a la cantidad de nodos total del árbol. Se realiza un cambio a la arquitectura original de Yolo 9000 reemplazando las capas de *Softmax* como fuente de información jerárquica por una optimización jerárquica basada en calcular la función de costo como la suma de las entropías cruzadas de las clases hijas de los nodos de decisión. Una representación de esta red puede ser visto en la Figura 4.11.

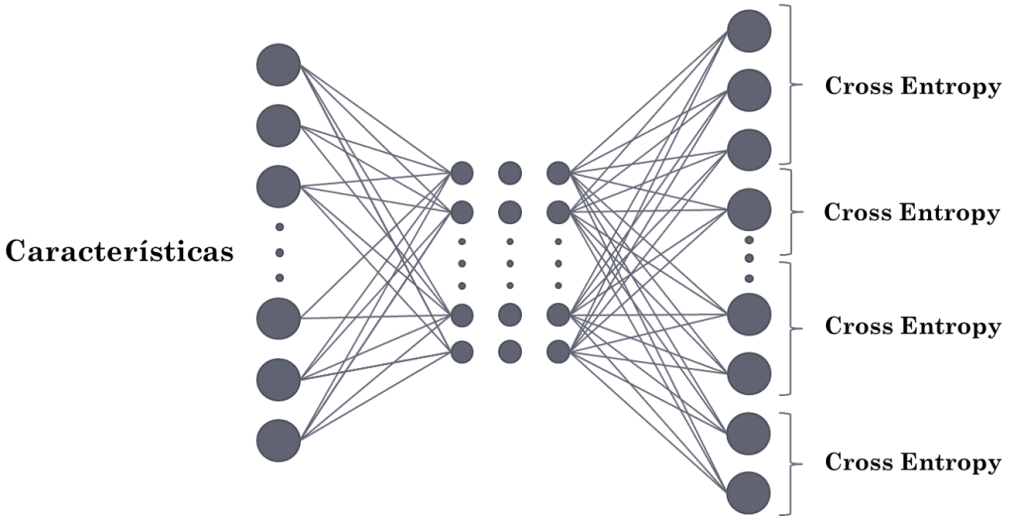


Figura 4.11: Arquitectura de la red HiGNet

Evaluación y comparación Los puntos en los que se hará hincapié para evaluación son:

- **Tiempos de ejecución**: Tiempos de entrenamiento, evaluación sobre el conjunto de test y sobre ejemplos individuales.
- **Métricas jerárquicas**: Precisión y recall jerárquico.
- **Métricas por nivel**: Precisión y recall de los niveles del árbol de categorías. En la ecuación 4.2 se puede ver las expresiones matemáticas correspondientes en función de

su nivel l.

$$Precision(l) = \frac{VP(l)}{PrediccionesTotales(l)} \quad Recall(l) = \frac{VP(l)}{TotalPositivos(l)} \tag{4.2}$$

Finalmente se analizan los enfoques tomando en consideración el desempeño de los clasificadores y sus tiempos de ejecución para escoger el más adecuado para resolver el problema.

Capítulo 5

Análisis y Resultados

5.1. Exploración de datos

Dentro de los primeros acercamientos hacia la creación de los clasificadores jerárquicos está el conocimiento y exploración de la **distribución de los datos** y de la **estructura del árbol** de categorías de la base de datos del marketplace.

Las primeras observaciones que se pueden obtener sobre la estructura del árbol de categorías son las siguientes:

- **Cantidad de clases de primer nivel:** 27 (ver Anexo Figura A.2)
- **Número de niveles:** 7
- **Cantidad de clases nodo:** 19342
- **Cantidad de clases nodo hoja:** 15126

La cantidad de datos descargados corresponde a **3.131.683** fichas con información de títulos y fotografías, luego estos datos son divididos en los conjuntos de entrenamiento (80 %), validación (20 % conjunto de entrenamiento) y test (20 %).

Con el fin de tener un mejor entendimiento de la distribución de datos se procede a graficar la cantidad de ejemplos de cada una de las 27 clases principales del árbol de categorías y su distribución puede ser vista en la Figura 5.1 para todos los conjuntos. En esta gráfica se observa como existen clases padres sobrerrepresentadas y esto se aprecia la Tabla 5.1 donde se muestran las cinco categorías con más elementos, las cuales suman el 42.4 % de los datos. Este desbalance puede generar problemas en la clasificación, ya que la red va a tender a predecir a las clases con más representación.

Label	Clase	Cantidad	Porcentaje
0	Accesorios para Vehículos	373.300	12 %
11	Deportes y Fitness	259.194	8,3 %
9	Computación	248.162	8 %
16	Hogar y Muebles	221.783	7,1 %
23	Relojes y Joyas	215.503	6,9 %
Total	-	1.317.942	42,4 %

Tabla 5.1: Cinco clases con más elementos

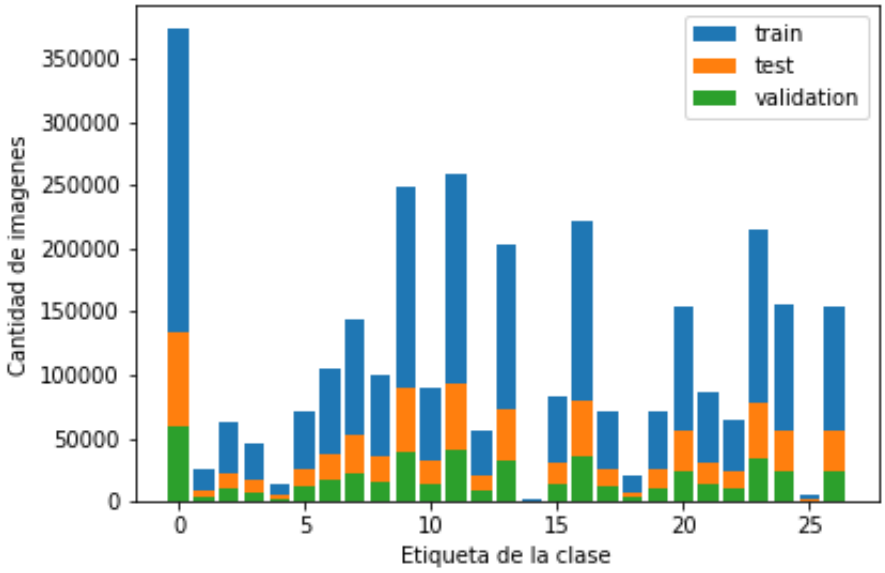


Figura 5.1: Distribución de productos sobre las 27 principales clases del árbol de categorías

5.2. Revisión de datos

Se revisa manualmente 500 fotos de cada una de las clases padres y se calcula el coeficiente I_{clase} para cada uno de ellos. En la Figura Anexo A.1 se pueden ver los resultados de cada una de las clases principales. En esta se puede observar que las categorías con mayor cantidad de etiquetas ruidosas son "Entradas para Eventos y Servicios" con I_{clase} de 0,47 y 0,49 respectivamente, mientras que los que poseen coeficiente más alto son Autos, Motos y Otros e Inmuebles con 0,96 y 0,95 respectivamente. Cabe destacar que solo cuatro de las clases poseen I_{clase} mayor a 0,9.

Un fenómeno observado en la revisión de los datos es el efecto del clasificador de texto sobre algunas etiquetas incorrectas, alguno de estos ejemplos pueden ser vistos en la Figura 5.2.

Imagen	Titulo	Etiqueta	Revisión
	Uco Candlelier Deluxe Linterna De Velas	Coleccionables Figuras de acción Super Héroes DC Linterna verde	
	Lente Armazon Óptico Dama Caballero Mytho	Cámaras y accesorios Instrumentos ópticos Lupas	

Figura 5.2: Ejemplos de etiqueta incorrecta. El primero producto corresponde a una linterna de velas para camping, pero debido a la predicción por texto su etiqueta final es Linterna Verde. El segundo producto corresponde a lentes ópticos, pero la predicción por texto lo deja en la categorías Lupas.

5.3. Extracción de características imágenes

Como fue mencionado en la sección 4.5 la red usada es una Resnet-18 pre-entrenada en ImageNet y tiene una salida de 1x2048. En este trabajo se hace un pre-procesamiento de las imágenes para que se adapten a la red y posteriormente una reducción de dimensionalidad a través de PCA. Estos procesos se detallan a continuación:

Transformación imágenes Las redes neuronales convolucionales típicamente reciben imágenes cuadradas, en particular la Resnet-18, que recibe imágenes de 224x224 pixeles, por lo que se debe pasar por una transformación para poder realizar las predicciones.

El framework de trabajo de redes neuronales elegido es *PyTorch* para el lenguaje de programación *Python*. Este provee de distintas transformaciones como las siguientes:

- Resize: Tomar la imagen y escalarla a una imagen cuadrada de las dimensiones apropiadas.
- Crop: Cortar un cuadrado del centro de la imagen de la dimensión requerida.
- Crop+resize: Escalar la imagen en su dimensión más pequeña a la dimensión requerida, por ejemplo, si altura > ancho, entonces la imagen será reescalada a (size*altura/ancho,size). Después de esto se aplica un recorte central sobre esta.

En la Figura 5.3 se puede ver que crop+resize conserva mejor las proporciones de los objetos en comparación.

Mientras tanto en la Figura 5.4 se puede observar que al tener fotografías muy alargadas

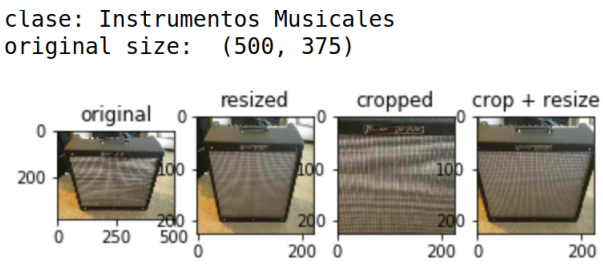


Figura 5.3: Foto de amplificador

crop+resize corta la imagen, mientras que resize puede obtener la imagen completa (sin respetar sus dimensiones).

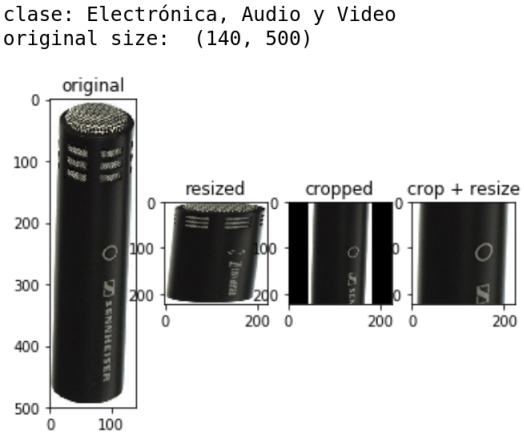


Figura 5.4: Foto de micrófono alargada

Finalmente se argumenta utilizar crop+resize, ya que si bien algunos objetos no aparecen completos se logra obtener parte parcial de este y se asume que podrá ser identificado.

Principal Component Analysis Se busca obtener una reducción de la dimensionalidad de los vectores a través del análisis PCA buscando bajar tiempos de entrenamiento y predicción, sin afectar la efectividad del clasificador.

Para esto se toma un conjunto aleatorio de 1 millón de vectores de características del conjunto de entrenamiento y se busca preservar el **95 %** de la variabilidad. Al finalizar este proceso se puede pasar de vectores de **2048 características a solo 462**.

5.4. Extracción de características de texto

La extracción de características se realiza utilizando *FastText*, que permite entrenar un modelo no-supervisado para luego ser utilizado como extractor de características. Este modelo es entrenado con los siguientes parámetros:

- **min n**: 2

- **max n**: 4
- **word n grams**: 3
- **learning rate**: 0.05
- **dimension**: 100
- **epoch**: 25

Después del entrenamiento se obtienen vectores de tamaño $1x100$.

5.5. Poda del árbol

El árbol de categorías en su forma original posee un total de 19.342 nodos y se busca reducir el tamaño de este usando los criterios y pasos descritos en la sección 4.7 y para esto se realiza lo siguiente:

Eliminación clase Otros Se parte eliminando todas las clases y su descendencia que contengan las palabras "Otros" u "Otras" en su nombre con excepción de la clase "Autos, Motos y Otros", debido a que es una clase principal. Un nodo de ejemplo de esto es "Otros Eventos' de la jerarquía de "Entradas para Eventos". Este proceso lleva a descartar 1.279 nodos reduciéndose así el tamaño del árbol a 18.063 nodos.

Eliminación clases ruidosas En este caso se considerará un umbral de 0,8 para el coeficiente I_{clase}, esto se considera para las clases padres y se eliminan todas aquellas que no cumplan con $I_{clase} < umbral$. Las clases seleccionadas bajan de 27 clases principales a 15 y pueden ser vistas en la Tabla 5.2. Este proceso descarta un total de 5.493 nodos reduciéndose el árbol de categorías a 12.570.

Mínimo de ejemplos El número mínimo de ejemplos determinado para que un nodo sea considerado como válido es fijado en 300 ejemplos de entrenamiento, esto para que los algoritmos tengan suficientes datos para ser entrenados. Este proceso es el que produce mayor reducción dentro del árbol con un total de 8.948 nodos descartados, reduciendo el árbol de categorías a 3.622 nodos.

Limitación de niveles La última consideración que se va tener en cuenta es la de limitar la cantidad de niveles del árbol, esto debido a las limitaciones computacionales para entrenar los algoritmos asociados a redes neuronales, ya que éstos escalan en complejidad con la cantidad de nodos del árbol, por lo tanto, todos los algoritmos son evaluados y entrenados con sólo los 4 primeros niveles del árbol. Esto produce que se descarten 718 quedando el árbol finalmente con 2.904 nodos.

Etiqueta	**Clase**	***I_ Clase***
0	Accesorios para Vehículos	0,85
1	Arte y Antigüedades	0,93
2	Autos, Motos y Otros	0,96
3	Bebés	0,81
4	Cámaras y Accesorios	0,84
5	Celulares y Telefonía	0,81
6	Coleccionables y Hobbies	0,88
7	Computación	0,86
8	Consolas y Videojuegos	0,81
9	Inmuebles	0,95
10	Juegos y Juguetes	0,87
11	Música y Peliculas	0,89
12	Relojes y Joyas	0,9
13	Salud y Belleza	0,81
14	Vestuario y Calzado	0,88

Tabla 5.2: Clases padres seleccionadas con I_{clase} mayor a 0.8

En la Tabla 5.3 se puede ver un resumen de las etapas de poda de nodos y la cantidad de nodos conservados después de la poda de nodos. Originalmente, el árbol de categorías posee un total de **19.342** nodos y después del proceso de poda este queda con **2.905**.

Fase	Nodos
Original	19.342
Eliminar clases Otros	18.063
Eliminar clases ruidosas	12.570
Eliminar nodos con min de ejemplos	3.622
Limitación de niveles	**2.904**

Tabla 5.3: Resumen poda de nodos

5.6. Comparación de clasificadores jerárquicos

Una vez entrenados los clasificadores jerárquicos con los tres tipos de descriptores: Imagen, texto y concatenados, se procede a realizar una evaluación sobre el conjunto de test. Este conjunto consiste en un total de **395.417** datos y en la Figura 5.5 se puede apreciar el decrecimiento de la cantidad de ejemplos a medida que se desciende por el árbol. Esto puede ser visto a más detalle en la Tabla 5.5 para los niveles que serán utilizados en este trabajo. Cabe destacar que las dimensiones obtenidas de los diferentes descriptores pueden ser vistas en la Tabla 5.4.

Característica	Texto	Imagen	Concatenado
Tamaño de vector	100	462	562

Tabla 5.4: Tamaño de los vectores de característica

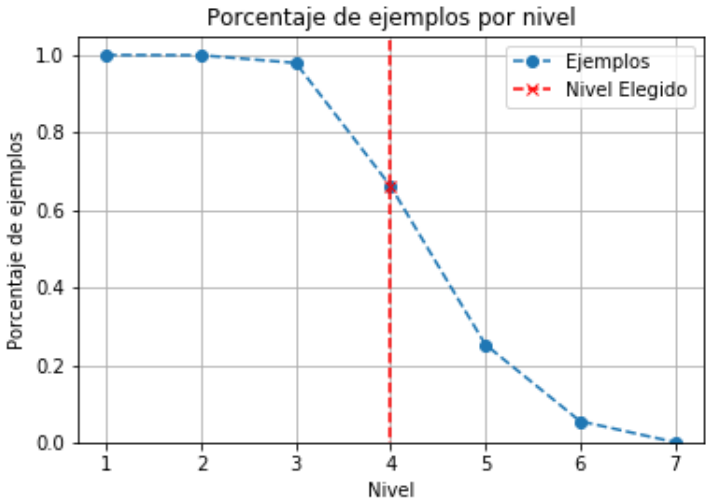


Figura 5.5: Porcentaje de ejemplos por nivel. Estos se obtiene dividendo los ejemplos presentes en el nivel por el total de ejemplos.

Nivel	1	2	3	4
Número de ejemplos	395.417	395.293	388.269	292.967
Porcentaje	100 %	99,9 %	98,2 %	74,1 %

Tabla 5.5: Número de ejemplos y porcentaje por nivel. Se representan los niveles después del proceso de poda del árbol.

Las Tablas 5.5, 5.6 y 5.7 poseen una nomenclatura en común, en donde:

- **H Precision**: Precisión jerárquica calculada con los métodos de la sección 2.7.2.
- **H Recall**: Recall jerárquico calculada con los métodos de la sección 2.7.2.
- **Train t**: Tiempo de entrenamiento en segundos.
- **Test t**: Tiempo de evaluación sobre el conjunto de test.
- **Ind. t**: Tiempo de evaluación sobre un ejemplo.

5.6.1. Caso 1: Clasificador multilabel

Como fue explicado en la sección 4.8, el caso base de este problema consiste en un clasificador multi-etiqueta sobre todos los nodos del árbol. Para esto se implementa una red neuronal tipo MLP para cada uno de los tipos de descriptores.

Arquitectura

- **Número de capas ocultas**: 4
- **Tamaño de capas ocultas**: 100
- **No linealidad capas oculta**: ReLU
- **No linealidad de salida**: Sigmoide
- **Tamaño de entrada**: 100, 462 y 562 que corresponden a texto, imágenes y concatenación (ver Tabla 5.4)
- **Tamaño de salida**: 2904

Entrenamiento El entrenamiento es llevado a cabo con **SGD** (*Stochastic Gradient Descent*) usando como función de pérdida **MSE** (*Mean Square Error*) durante 4 épocas con *learning rate* inicial de 10^{-3}. Este proceso tarda 11 horas para el descriptor más liviano de texto y 13 horas para la concatenación de ambos.

Resultados La Figura 5.6 presenta los resultados para los tres tipos de descriptores del clasificador multi-label. Para este tipo de clasificador se observa que las características de imagen contienen mayor información jerárquica que las de texto, pero a su vez estas últimas son mejores para clasificar en el primer nivel. El mejor resultado fue obtenido con la combinación de ambas características. Finalmente se puede apreciar un resumen del desempeño en la Tabla 5.6, en donde si bien los tiempos de ejecución para ejemplos individuales son bajos, los tiempos de prueba sobre el conjunto completo de test son muy altos, llegando a los 51 minutos con 37 segundos para su versión de vectores concatenados.

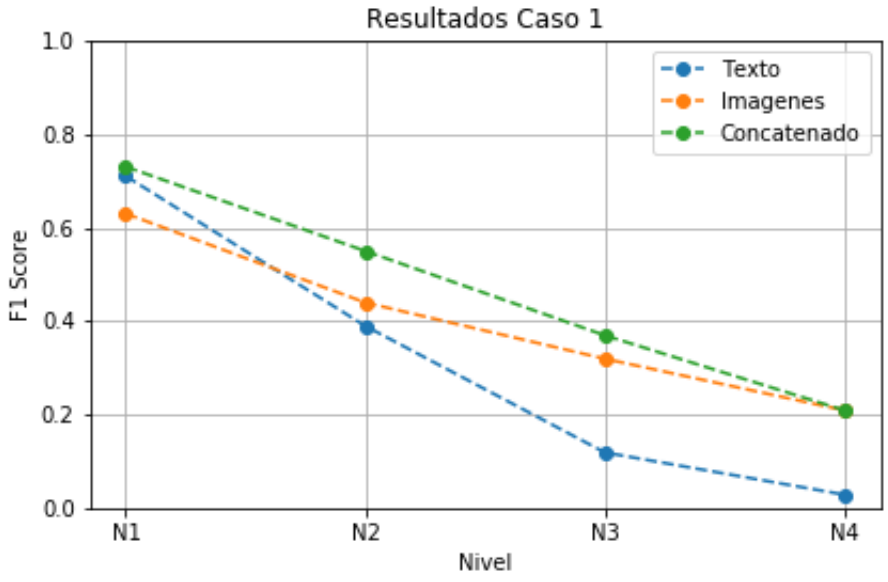


Figura 5.6: Comparación de F1 score por nivel de los distintos descriptores para el caso 1.

Características	H_Precision	H_Recall	Train t	Test t	Ind. t
Texto	0.24	0.25	11h 11m 06s	50m 47s	0.16s
Imágenes	0.36	0.37	13h 07m 32s	34m 58s	0.16s
Concatenado	**0.44**	**0.44**	**13h 30m 31s**	**51m 37s**	**0.16s**

Tabla 5.6: Resumen caso base

5.6.2. Caso 2: Clasificador jerárquico local

Para la construcción de este clasificador se implementa un modelo *Random Forest* en cada nodo de decisión del árbol.

Arquitectura

- **Número de árboles de decisión**: 100
- **Cantidad de clasificadores**: 628
- **Tamaño de entrada**: 100, 462 y 562 que corresponden a texto, imágenes y concatenación.

Entrenamiento El entrenamiento es llevado a cabo de manera jerárquica desde las hojas a la raíz con los métodos explicados en la sección 4.8.

Resultados La Figura 5.7 corresponde a los resultados del clasificador jerárquico local. En ésta se puede ver que el clasificador por características de texto es el que obtiene mejor resultado en la métrica F1. Además, conserva bien los detalles incluso en el último nivel, teniendo un desempeño de 0.762. En cuanto a las características concatenadas, se observa que la incorporación del vector de imágenes afecta negativamente al rendimiento del clasificador, obteniendo peores resultados con la utilización de texto por sí solo. En la Tabla 5.7 se puede ver que si bien los tiempos de ejecución en todo el conjunto de test son bajas (entre 5 y 9 minutos), sus tiempos de evaluación individual son muy altos, llegando incluso a un minuto para el caso de la concatenación de características.

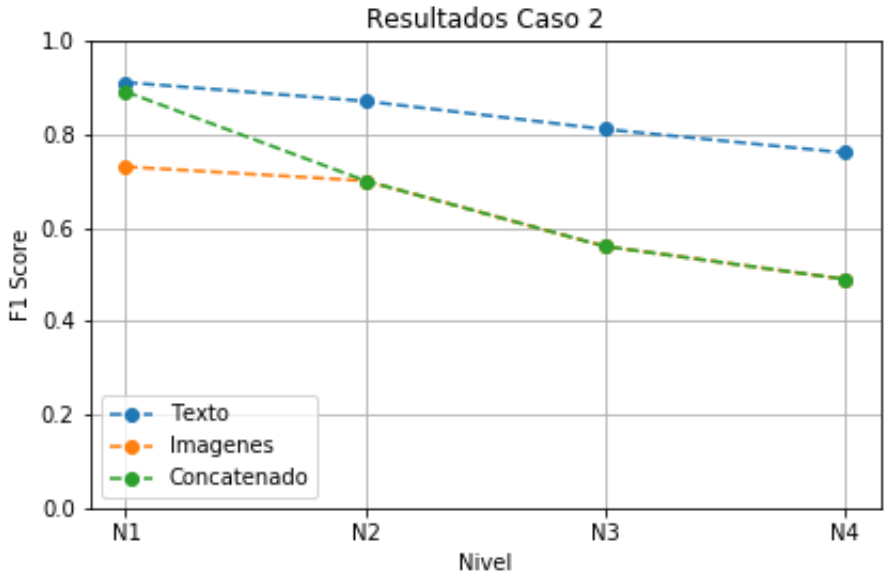


Figura 5.7: Comparación de F1 score por nivel de los distintos descriptores para el caso 2.

Características	H_Precision	H_Recall	Train t	Test t	Ind. t
Texto	**0.86**	**0.85**	**01h 54m 34s**	**05m 36s**	**9.3s**
Imágenes	0.59	0.61	04h 36m 25s	08m 22s	53s
Concatenado	0.66	0.68	04h 48m 57s	08m 30s	64s

Tabla 5.7: Resumen clasificado jerárquico local.

5.6.3. Caso 3: Clasificador jerárquico global

Para construir este clasificador se utiliza una arquitectura tipo MLP con las siguientes características:

Arquitectura

- **Número de capas ocultas**: 4
- **Tamaño de capas ocultas**: 100
- **No linealidad capas oculta**: ReLU
- **No linealidad de salida**: Identidad
- **Tamaño de entrada**: 100, 462 y 562 que corresponden a texto, imágenes y concatenación.
- **Tamaño de salida**: 2904

Entrenamiento El entrenamiento es llevado a cabo con **SGD** (*Stochastic Gradient Descent*) usando como función de pérdida de entropía cruzada jerárquica durante 4 épocas con

learning rate inicial de $2 * 10^{-3}$. Este proceso tarda 12 horas y 40 minutos para el descriptor más liviano de texto y 14 horas y 50 minutos para la concatenación de ambos.

Resultados La Figura 5.8 presenta los resultados para los tres tipos de descriptores del clasificador jerárquico global. Para este tipo de clasificador se observa que las características de imagen y texto funcionan complementariamente en el clasificador con características mixtas, obteniendo mejores resultados que su contra parte individual. Finalmente se puede apreciar un resumen del desempeño en la Tabla 5.8, en donde se puede apreciar que si bien los tiempos de ejecución para ejemplos individuales son bajos, los tiempos de prueba sobre el conjunto completo de test son muy altos, llegando a los 50 minutos para su versión de vectores concatenados.

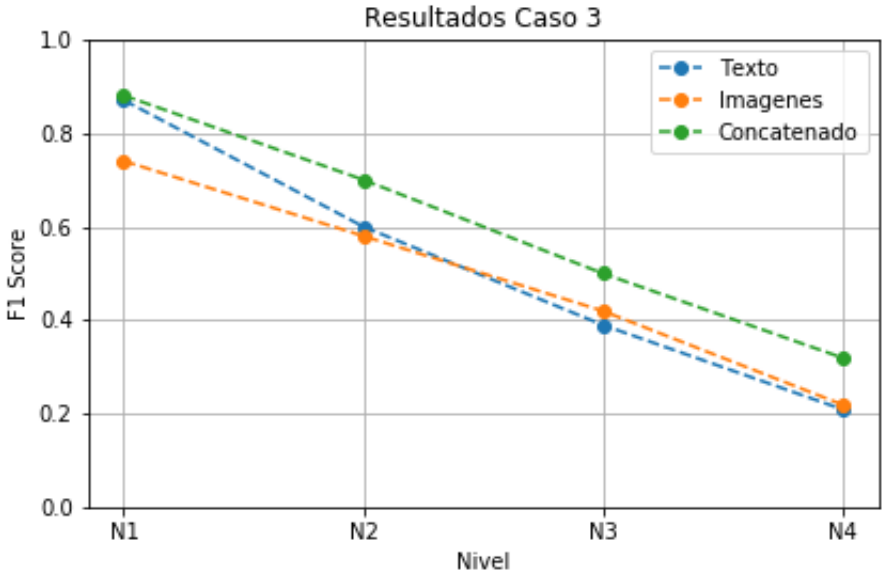


Figura 5.8: Comparación de F1 score por nivel de los distintos descriptores para el caso 3.

Características	**H_Precision**	**H_Recall**	**Train t**	**Test t**	**Ind. t**
Texto	0.50	0.50	12h 36m 07s	48m 15s	0.16s
Imágenes	0.47	0.47	13h 53m 22s	48m 38s	0.16s
Concatenado	**0.64**	**0.64**	**14h 50m 31s**	**50m 05s**	**0.16s**

Tabla 5.8: Resumen clasificado jerárquico global

5.6.4. Comparaciones

Se seleccionaron los mejores clasificadores de cada uno de los algoritmos propuestos para luego ser comparados entre sí y estos fueron:

- **Caso 1: Concatenado**
- **Caso 2: Texto**
- **Caso 3: Concatenado**

En la Figura 5.9 se puede apreciar la comparación de los mejores clasificadores para cada uno de los casos. En ésta se puede ver cómo el caso 1 es mucho más robusto a niveles más específicos dentro del árbol al compararlo con los otros dos clasificadores. Cabe destacar que el caso 3 que plantea un entrenamiento jerárquico presenta mejoras con respecto al caso 1 sin información alguna sobre el árbol de categorías.

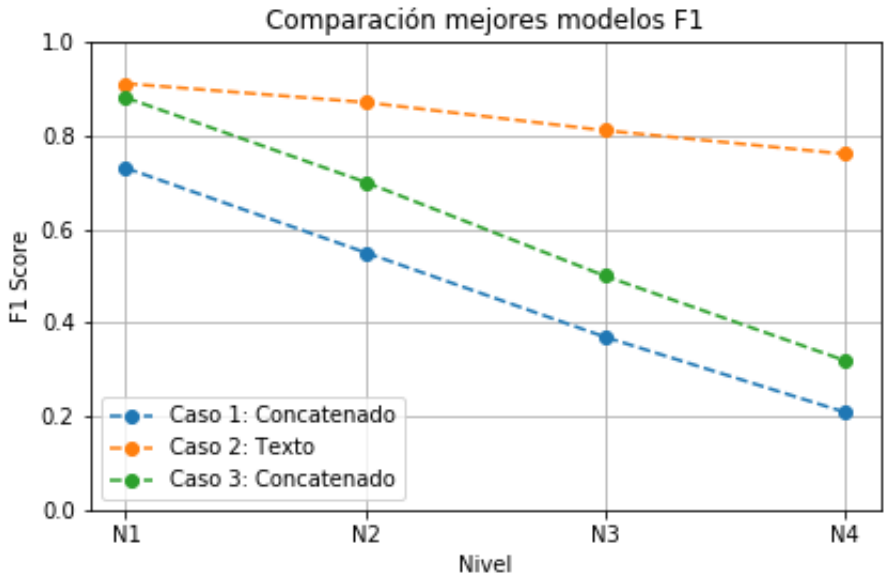


Figura 5.9: Comparación de métrica F1 por nivel de los mejores modelos.

Con respecto a los tiempos de ejecución de estos clasificadores, se puede apreciar en la Figura 5.10 que el caso 2 posee un tiempo de ejecución más prolongado para un ejemplo individual con 9.3 segundos. Mientras que los otros dos clasificadores se demoran 0.16 segundos cada uno.

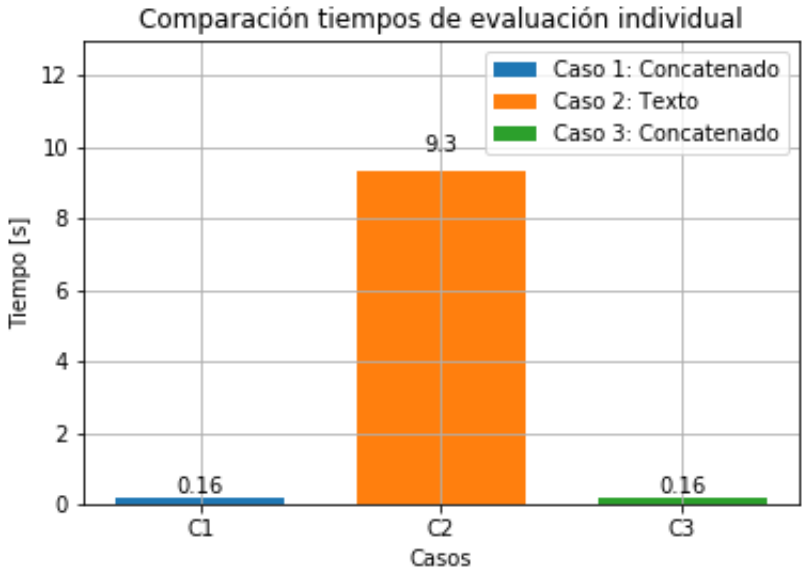


Figura 5.10: Comparación de tiempo de ejecución en evaluación de ejemplos individuales de los mejores modelos.

En las Figuras 5.11, 5.12, 5.14 y 5.13 se muestran los resultados de los distintos casos en fichas reales.

Ficha	Label	Caso 1	Caso 2	Caso 3	Nivel
Guantes Moto 100% Cuero 	Accesorios para Vehículos	Accesorios para Vehículos ✗	Autos, Motos y Otros ✗	Accesorios para Vehículos ✓	1
	Acc. para Motos y Cuatrimotos Indumentaria y Calzado	Limpieza de Vehículos ✗	Motos de Calle ✗	Acc. para Motos y Cuatrimotos Indumentaria y Calzado ✓	2
	Guantes	Desengrasantes ✗	Ducati ✗	Cubre puños ✗	3
	-	-	-	-	4

Figura 5.11: Resultado obtenido para tres casos. En este caso se aprecia que el clasificador por texto ante la presencia de la palabra “Moto” en el título se cataloga de manera errónea dentro de “Autos, Motos y Otros”, por otro lado, el clasificador correspondiente al caso 3 es robusto a este problema debido a la incorporación de la información otorgada por la imagen

Ficha	Label	Caso 1	Caso 2	Caso 3	Nivel
Bota De Arnés De Junction Yukon Para Mujer De Woolrich, M 	Vestuario y Calzado	Accesorios para Vehículos ✗	Accesorios para Vehículos ✗	Vestuario y Calzado ✓	1
	Calzados	Herramientas ✗	Accesorios de Auto y Camioneta ✗	Calzados ✓	2
	Botas	Cargadores ✗	Accesorios de Exterior ✗	Botas ✓	3
	Mujer	- ✗	Bota Agua ✗	Mujer ✓	4

Figura 5.12: Resultado obtenido para tres casos. En este ejemplo el clasificador de texto clasifica erróneamente una bota de mujer como una bota de agua para Accesorios para Vehículo, esto puede ser debido a la sobrerrepresentación de esta clase en el conjunto de entrenamiento, por otro lado, el caso 3 logra obtener la jerarquía correcta de manera completa debido al uso de la imagen.

Ficha	Label	Caso 1	Caso 2	Caso 3	Nivel
Base Notebook Ventilador 17 Data-com /3gmarket 	Computación	Consolas y Videojuegos ✗	Computación ✓	Consolas y Videojuegos ✗	1
	Notebooks y Accesorios	Consolas ✗	Notebooks y Accesorios ✓	Consolas ✗	2
	Accesorios para Notebooks	Pro ✗	Accesorios para Notebooks ✓	Premium ✗	3
	Bases Enfriadoras	- ✗	Bases Enfriadoras ✓	- ✗	4

Figura 5.13: Resultado obtenido para tres casos. En este ejemplo el clasificador por texto obtiene la jerarquía correcta de manera completa, pero los otros clasificadores clasifican incorrectamente la categoría debido al uso de la imagen.

Ficha	Label	Caso 1	Caso 2	Caso 3	Nivel
Apple iPhone 7 4g Teléfono Móvil Máquina Restaurada 128gb 	Celulares y Telefonía	Celulares y Telefonía ✓	Celulares y Telefonía ✓	Celulares y Telefonía ✓	1
	Celulares y Smartphones	Celulares y Smartphone ✓	Celulares y Smartphone ✓	Celulares y Smartphone ✓	2
	Apple	Apple ✓	Apple ✓	Apple ✓	3
	-	-	-	-	4

Figura 5.14: Resultado obtenido para tres casos. En este ejemplo todos los clasificadores funcionan de manera correcta, ya que tanto el titulo como la imagen son tienen información representativa de la jerarquía del producto.

Capítulo 6

Conclusiones y Trabajo a Futuro

6.1. Conclusiones

- Como producto final de esta memoria se obtiene: la construcción de una base de datos jerárquica con los datos extraídos del marketplace *MercadoLibre*, el entrenamiento y evaluación de un clasificador jerárquico local que utiliza clasificadores RandomForest como nodo de decisión dentro del árbol, la creación y evaluación de un clasificador jerárquico global bautizado como HiGNet (Hierarchical Global Network) que fue entrenado con una función de pérdida jerárquica. Finalmente estos clasificadores fueron comparados juntos a un caso base utilizando las métricas F1 por nivel y Precision/Recall jerárquicas y los tiempos de entrenamiento, test y evaluaciones individuales.
- Con respecto a la base de datos se ve que al tratarse de un problema real, ésta posee etiquetados ruidosos y es difícil estimar la tasa de etiquetado incorrecto sin una revisión intensiva por parte de un humano. A pesar de esto, se realiza una revisión manual parcial de las imágenes de la base de datos (ver Figura A.1), en donde se muestra que esta base de datos posee grandes problemas de etiquetados ruidosos en algunas de sus clases, por ejemplo en el caso de Electrodomésticos, con el 46 % de sus datos comprometidos. Esto puede influir en el desempeño de los clasificadores y con el fin de obtener mejores resultados se debe realizar una limpieza manual.
- Existe un notorio desbalance sobre las clases padres del árbol de categorías (ver Figura 5.1) llegando a que solo 5 clases principales posean el 42 % de los datos. Esto puede generar un sesgo en la predicción de los clasificadores, por ejemplo, en la Figura 5.12 se observa que el clasificador multilabel, al no tener seguridad sobre la clase real, predice la clase Accesorios para Vehículos, que corresponde a la clase más sobrerrepresentada de la distribución.
- La utilización de un *loss* jerárquico para entrenar el clasificador jerárquico global logra obtener un aumento del **20 %** en la métrica F1 en el nivel 1 y un **46 %** de aumento para el nivel 4 comparado con el caso base, lo cual demuestra la utilidad de incorporar la estructura del árbol dentro del aprendizaje de la red. Cabe destacar que prácticamente no existe una diferencia significativa de tiempo de evaluación entre ambos clasificadores.
- Con respecto a la comparación de los mejores modelos, se pudo observar una tendencia

de los algoritmos basados en redes neuronales a presentar un mejor desempeño con la combinación de las fuentes de información de texto e imágenes (ver Figuras 5.6 y 5.8). Mientras que, por otro lado, el clasificador local que utiliza *Random Forest* logra un mejor rendimiento con las características asociadas a texto y la concatenación del vector de imágenes degrada el desempeño (ver Figura 5.7). Esto puede ser atribuido a un sesgo de la base de datos a la información de texto, ya que esta base de datos utiliza la clasificación por texto para el orden de sus productos dentro del catalogo. Un indicio de esto puede ser visto en los ejemplos expuestos en la Figura 5.2.

- El clasificador con mejor resultado es el presentado en el caso 2, que corresponde a un clasificador jerárquico local con fuentes de datos por texto. La comparación puede ser vista en la Figura 5.9, donde se aprecia una clara diferencia a medida que se avanza en los niveles del árbol. El segundo con mejor desempeño corresponde al caso 3.
- Si bien el caso 2 posee los mejores resultados dada la métrica F1 por nivel, también fue el que presentó un mayor tiempo al comparar lo tres modelos con respecto a una evaluación individual, llegando a tener una diferencia de **58 veces** (ver Figura 5.10). Esto puede ser asociado a la implementación actual del mismo, en donde cada vez que se evalúa el modelo se llama desde disco cada uno de los clasificadores involucrados. Una solución a este problema sería trabajar con todos los clasificadores en memoria en todo momento, asumiendo el costo asociado a recursos computacionales.
- La utilización del caso 3 es una alternativa liviana de un solo clasificador al caso 1, donde se deben utilizar 628 clasificadores. Si bien la pérdida de desempeño para la métrica F1 en el primer nivel es reducida (de 0.907 a 0.884), para niveles superiores aumenta bastante, llegando a ser un 42 % peor en el nivel 4.
- Finalmente la elección del clasificador **depende** del uso y recursos que disponga la plataforma: si es que esta posee una cantidad de recursos computacionales suficientes, se recomienda la utilización de un clasificador local jerárquico por texto, ya que en este trabajo ha demostrado ser robusto a los niveles más específicos del árbol. Por otro lado, si no se cuenta con estos recursos, se recomienda utilizar el clasificador jerárquico global con la combinación de descriptores de imagen y texto y limitar la cantidad de niveles sobre los cuales hacer predicciones, por ejemplo, realizar predicciones sobre los primeros 2 o 3 niveles, en donde este clasificador tiene alta precisión.

6.2. Trabajo a Futuro

- Una práctica común para aumentar el rendimiento de los algoritmos de tipo red neuronal es el entrenamiento end-to-end, que consiste en entrenar tanto el extractor de características como el clasificador en una sola red, con el fin de encontrar el mejor espacio de características para la tarea que se busca resolver. En este caso se espera que los resultados mejoren al utilizar redes especializadas tanto para texto como para imágenes.
- La forma de unión de características utilizada en este trabajo es considerada una de las más simples para incorporar información de diferentes fuentes. Es por esto que se propone usar formas más elaboradas de unión de características para obtener mejores resultados. Una alternativa a esto puede ser el entrenamiento de un regresor que con-

vierta las características de imagen al espacio de texto y luego concatenar ambos, con el fin de complementar la información.

- Finalmente se propone utilizar las mismas técnicas descritas en este trabajo en una nueva base de datos asociada a un e-commerce real que posea las estructuras de ficha y de árbol descritas anteriormente.

Apéndice A

Anexo A

Etiqueta	**Clase**	***I_ Clase***
0	Accesorios para Vehículos	0,85
1	Alimentos y Bebidas	0,64
2	Animales y Mascotas	0,62
3	Arte y Antigüedades	0,93
4	Autos, Motos y Otros	0,96
5	Bebés	0,81
6	Cámaras y Accesorios	0,84
7	Celulares y Telefonía	0,81
8	Coleccionables y Hobbies	0,88
9	Computación	0,86
10	Consolas y Videojuegos	0,81
11	Deportes y Fitness	0,69
12	Electrodomésticos	0,54
13	Electrónica, Audio y Video	0,71
14	Entradas para Eventos	0,47
15	Herramientas y Construcción	0,57
16	Hogar y Muebles	0,59
17	Industrias y Oficinas	0,65
18	Inmuebles	0,95
19	Instrumentos Musicales	0,78
20	Juegos y Juguetes	0,87
21	Libros, Revistas y Comics	0,78
22	Música y Películas	0,89
23	Relojes y Joyas	0,9
24	Salud y Belleza	0,81
25	Servicios	0,49
26	Vestuario y Calzado	0,88

Tabla A.1: Estimación del coeficiente I_{clase} en las clases principales del árbol de categorías.

Etiqueta	**Clase**
0	Accesorios para Vehículos
1	Alimentos y Bebidas
2	Animales y Mascotas
3	Arte y Antigüedades
4	Autos, Motos y Otros
5	Bebés
6	Cámaras y Accesorios
7	Celulares y Telefonía
8	Coleccionables y Hobbies
9	Computación
10	Consolas y Videojuegos
11	Deportes y Fitness
12	Electrodomésticos
13	Electrónica, Audio y Video
14	Entradas para Eventos
15	Herramientas y Construcción
16	Hogar y Muebles
17	Industrias y Oficinas
18	Inmuebles
19	Instrumentos Musicales
20	Juegos y Juguetes
21	Libros, Revistas y Comics
22	Música y Películas
23	Relojes y Joyas
24	Salud y Belleza
25	Servicios
26	Vestuario y Calzado

Tabla A.2: Labels y nombre de clases padres.

Apéndice B

Anexo B

En este anexo se presentan los resultados por nivel de los distintos clasificadores con los tres tipos de características.

B.1. Caso 1: Clasificador multilabel

Level	Level Precision	Level Recall	Level F1
1	0,712	0,712	0,712
2	0,387	0,402	0,394
3	0,114	0,143	0,126
4	0,019	0,037	0,025

Tabla B.1: Resultado por nivel de clasificador multilabel para características de **texto**.

Level	Level Precision	Level Recall	Level F1
1	0,627	0,627	0,627
2	0,429	0,432	0,430
3	0,313	0,331	0,322
4	0,197	0,227	0,221

Tabla B.2: Resultado por nivel de clasificador multilabel para características de **imágenes**.

Level	Level Precision	Level Recall	Level F1
1	0,733	0,733	0,733
2	0,536	0,559	0,547
3	0,372	0,379	0,375
4	0,212	0,225	0,218

Tabla B.3: Resultado por nivel de clasificador multilabel para características **concatenadas**.

B.2. Caso 2: Clasificador jerárquico local

Level	Level Precision	Level Recall	Level F1
1	0,907	0,907	0,907
2	0,857	0,868	0,862
3	0,812	0,816	0,814
4	0,718	0,813	0,762

Tabla B.4: Resultado por nivel de clasificador jerárquico local para características de **texto**.

Level	Level Precision	Level Recall	Level F1
1	0,783	0,783	0,783
2	0,703	0,704	0,703
3	0,548	0,579	0,563
4	0,417	0,591	0,489

Tabla B.5: Resultado por nivel de clasificador jerárquico local para características de **imágenes**.

Level	Level Precision	Level Recall	Level F1
1	0,893	0,893	0,893
2	0,697	0,701	0,699
3	0,547	0,576	0,561
4	0,419	0,592	0,491

Tabla B.6: Resultado por nivel de clasificador jerárquico local para características **concatenadas**.

B.3. Caso 3: Clasificador jerárquico global

Level	Level Precision	Level Recall	Level F1
1	0,867	0,867	0,867
2	0,598	0,598	0,598
3	0,393	0,397	0,395
4	0,209	0,223	0,216

Tabla B.7: Resultado por nivel de clasificador jerárquico global para características de **texto**.

Level	Level Precision	Level Recall	Level F1
1	0,737	0,737	0,737
2	0,585	0,590	0,587
3	0,417	0,423	0,420
4	0,207	0,227	0,216

Tabla B.8: Resultado por nivel de clasificador jerárquico global para características de **imágenes**.

Level	Level Precision	Level Recall	Level F1
1	0,884	0,884	0,884
2	0,704	0,712	0,708
3	0,503	0,505	0,504
4	0,317	0,323	0,320

Tabla B.9: Resultado por nivel de clasificador jerárquico global para características **concatenadas**.

Bibliografía

[1] Piotr Bojanowski Tomas Mikolov. Armand Joulin, Edouard Grave. Bag of tricks for efficient text classification. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, 2016.

[2] L. Schietgat S. Dzeroski andH. Blockeel. C. Vens, J. Struyf. Decision trees for hierarchical multi-label classication. machine learning, 73:185–214, 2008.

[3] Dong W. Socher R. Li L.-J. Li K. Deng, J. and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. *CVPR*, 2009.

[4] digitalcommerce360. e-commerce ten year review. `https://www.digitalcommerce360.com/article/e-commerce-sales-retail-sales-ten-year-review/`, 2020.

[5] C. Fellbaum D. Gross G. A. Miller, R. Beckwith and K. J. Miller. Introduction to wordnet: An on-line lexical database. *International journal of lexicography*, 3(4):235–244, 1990.

[6] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks, 4(2), 251–257*, 1991.

[7] Ali Farhadi. Joseph Redmon. Yolo9000:better, faster, stronger. *CVPR*, 2017.

[8] Shaoqing Ren Jian Sun Kaiming He, Xiangyu Zhang. Deep residual learning for image recognition. *CVPR*, 2015.

[9] Sutskever I. Krizhevsky, A. and G.E. Hinton. Imagenet classification with deep convolutional neural networks. *NIPS*, 2012.

[10] MercadoLibre. Mercadolibre api. `https://developers.mercadolibre.cl/`, 2020.

[11] C. Silla and A. Freitas. A survey of hierarchicalclassication across dierent application domains.data mining and knowledge discovery, 22:31–72,2010.

[12] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CVPR*, 2014.

[13] S. Belongie J. Hays P. Perona D. Ramanan P. Dollar T.-Y. Lin, M. Maire and C. L. Zitnick. Microsoft coco: Common objects in context. *ECCV*, 2014.