



UNIVERSIDAD DE BUENOS AIRES
FACULTAD DE CIENCIAS EXACTAS Y NATURALES
DEPARTAMENTO DE COMPUTACIÓN

Modelos de redes convolucionales en grafos para la predicción de sabores y olores a partir de moléculas

Tesis de Licenciatura en Ciencias de la Computación

Tomás Jaratz

Director: Dr. Facundo Carrillo

Codirector: Ing. Federico Zamberlan

Buenos Aires, 2022

RESUMEN

A diferencia de la visión y la audición, el sentido del olfato sigue siendo poco comprendido, difícil de modelar e incluso más difícil de predecir[21]. A pesar de haber sido una herramienta evolutiva vital para la supervivencia, permitiéndonos diferenciar los alimentos en malas condiciones y detectar incendios cercanos[27], todavía no tenemos forma de predecir cómo olerá un compuesto químico a partir de conocer su estructura molecular[21][38]. Al igual que ocurre con los fármacos, las moléculas responsables de que percibamos estas sensaciones mediante nuestros sentidos también se acoplan a proteínas receptoras transmembrana en la superficie de las células del sistema nervioso[24], en este caso, ubicadas en nuestra nariz o boca. Actualmente, a excepción de algunos grupos de moléculas específicos, la única forma de saber cómo va a oler una sustancia es oliéndola, ya que no existe un método general para inferir este tipo de información en base a las propiedades de la molécula . El desarrollo de un modelo predictivo que trabaje sobre este fenómeno de interacción entre proteínas y ligandos resulta una herramienta de vital importancia con inmediata y directa transferibilidad a otros problemas de la industria química[37], desde el desarrollo de medicamentos[39] hasta saborizantes[32] o insecticidas[34]. El presente trabajo tiene por objetivo el desarrollo de modelos que permitan determinar a partir de la estructura molecular de una sustancia las etiquetas que mejor describan sus características olfativas y gustativas. Para esto, trabajaremos en la implementación y uso de redes de convolución para grafos, una arquitectura de aprendizaje profundo que se encuentra en auge y demostrando mucho potencial para trabajar con datos no estructurados. Nuestra hipótesis es que las moléculas que comparten características olfativas y gustativas similares deben también presentar similitudes en la disposición geométrica de sus átomos en el espacio. Al codificar la estructura molecular a través de grafos con los atributos de los átomos en los nodos estamos capturando una amplia información sobre la estructura molecular. Finalmente las redes de grafos convolucionales deberían ser capaces de hallar las relaciones entre dicha geometría y el efecto biológico subjetivo provocado por las moléculas.

Palabras clave: perceptrón multicapa, redes de grafos convolucionales, FlavorDB, molécula, olor, gusto, ROC, AUC.

AGRADECIMIENTOS

- A Pablo, Magda y Santi.
- A mis mejores amigos y al único que me acompañó desde el primer hasta el último día, Toti.
- A la Universidad de Buenos Aires, al DC, sus profesores y ayudantes.
- A Fede Pousa y Pablo Riera por ser los grandes profesores y jurados que fueron.
- A Facu Carrillo por dirigir esta tesis.
- A Fede Zamberlan, por haberme guiado en todo momento y seguir haciéndolo. Y principalmente, por ser un gran amigo.
- A Enzo Tagliazucchi y a todo COCUCO, por todo el apoyo que me dieron y por producir este trabajo. Pero sobre todo, por enseñarme a dar las gracias.
- A mi familia.

Gracias

Y perdón, los debería haber incluído antes.

Índice general

1	Introducción	1
1.1.	Background y definición del problema	1
1.2.	Las moléculas	2
1.3.	El sabor y su relación con el gusto y el olfato	4
1.4.	Grafos	7
1.5.	Redes Neuronales Artificiales	11
2	Obtención y estructuración de los datos	14
2.1.	Flavor DB	14
2.2.	Explorando los tags	15
3	Preprocesamiento, de InChis a grafos	18
3.1.	InChis	18
3.2.	RdKit	20
3.3.	Features	21
3.4.	Pasaje a grafo	25
4	Redes de Grafos convolucionales	27
4.1.	Redes convolucionales	28
4.2.	Convoluciones en grafos	31
5	Experimentación	39
5.1.	Software y Metodología	39
5.2.	Dulce vs Amargo	42
5.3.	Multiclasificación de sabores	46
5.4.	Detección de moléculas con olor	51
6	Conclusiones y trabajo futuro	54
	Bibliografía	56

1. INTRODUCCIÓN

1.1. Background y definición del problema

La percepción de los sabores es un fenómeno causado por la interacción de ciertas moléculas al entrar en contacto con las células que se encuentran en las papilas gustativas y el tracto olfativo generando estímulos nerviosos que se propagan hasta el cerebro donde son procesados e interpretados como sensaciones. El olfato y el gusto, responsables de procesar los sabores, son los sentidos menos comprendidos ya que no resulta del todo claro que factores determinan que un compuesto tenga un sabor particular. Industrias como la perfumística y alimentaria se encuentran en constante búsqueda de nuevos compuestos con olores y gustos específicos, de mayor o menor intensidad o duración, para esta labor los químicos necesitan comprender la relación entre el sabor percibido y las propiedades físico-químicas de un compuesto. El proceso de descubrimiento de nuevas moléculas con características específicas no suele ser simple y muchas veces es costoso ya que uno de los principales inconvenientes que atraviesan estas industrias radica en que luego de hipotetizar sobre las posibles propiedades de un nuevo compuesto, es condición necesaria sintetizarlo para saber si cumple con las características buscadas o debe ser descartado en pos de continuar con la búsqueda. En el caso del olfato por ejemplo, para que una molécula tenga olor debe ser lo suficientemente volátil como para ser transportada por el aire y llegar a la cavidad nasal, debe tener un bajo peso molecular, menor a 500 Da y tener cierto grado de hidrofobicidad (propiedad que repele al agua), al mismo tiempo debe ser agonista de algún receptor de la cavidad nasal de modo que pueda generar el estímulo nervioso que se propaga hasta el cerebro.

Si bien han habido múltiples estudios dedicados a encontrar patrones estructurales en las moléculas que describan su sabor, salvo por algunos grupos reducidos como los ésteres o los aldehídos, no existe un mecanismo establecido que permita derivar el sabor de una molécula en base a sus propiedades químicas. Esto nos introduce a un problema de índole filosófico subyacente a la inteligencia artificial y la representación del conocimiento, en donde existe una relación desconocida entre la percepción de un fenómeno, el sabor y los olores, la naturaleza de su estructura, las moléculas, y las leyes físicas que rigen el modo en que los percibimos distintos o no los percibamos en absoluto. En este sentido, los métodos de aprendizaje automático cumplen un rol fundamental al proveer algoritmos capaces de hallar patrones y relaciones en espacios y estructuras matemáticas sobre los que se codifican atributos del fenómeno que se quiere modelar, usualmente estos patrones suelen ser casi imposibles de deducir mediante otras técnicas. En este trabajo abordamos específicamente el modelado de las moléculas y sus propiedades con el fin de crear programas, basados en técnicas de deep learning, que sean capaces de aprender a distinguir propiedades del sabor, como pueden ser la amargura o el dulzor, entre otros, de nuevas moléculas nunca antes vistas. Este tipo de tecnologías pueden resultar muy útiles para indicar con cierto grado de probabilidad si un compuesto va a tener un sabor en específico o ninguno en absoluto, de forma de mejorar el espacio de búsqueda sobre el cual los científicos deciden qué compuestos sintetizar y cuáles no.

Para modelar las moléculas utilizamos una estructura matemática llamada **grafo**, la cual permite representar las relaciones que existen entre los átomos que componen la mo-

lécula y su estructura en general, luego construimos y experimentamos con una serie de arquitecturas neuronales llamadas **redes de grafos convolucionales**, una tecnología en auge que viene siendo explorada en los últimos años y está especialmente dedicada a procesar información cuya estructura puede describirse mediante grafos, de ahí el nombre “redes de grafos convolucionales”. Para comprenderlas definiremos la arquitectura y el funcionamiento subyacente de este tipo de redes y algunas de sus variantes, así como las propiedades de las moléculas que fueron modeladas. Finalmente pusimos a prueba los modelos mediante experimentos utilizando ejemplos tomados de un repositorio de moléculas y sus sabores llamado **FlavorDB** y tratamos de comprender si los modelos fueron capaces de capturar patrones estructurales en la información provista

1.2. Las moléculas

Para comprender las moléculas, es necesario introducir primero un concepto más primitivo y fundamental, el **átomo**. La palabra “átomo”, deriva del griego *átomon* (aquel que no se puede cortar, que es indivisible), se refiere a la unidad de materia más pequeña del universo que posee propiedades químicas. Son los elementos microscópicos que componen la materia sólida, líquida, gaseosa y de plasma. Los átomos a su vez se componen de partículas más pequeñas, en su centro llamado núcleo se encuentran los neutrones y los protones (partículas con carga positiva), y en su exterior se encuentran los electrones (partículas de carga negativa) orbitando el núcleo y atraídos a este por las fuerzas electromagnéticas que se dan entre las cargas de estos y de los protones. El número de protones en el núcleo es el que define que elemento químico es el átomo en cuestión, mientras que la cantidad y la disposición de los electrones influyen en las propiedades magnéticas del átomo. A su vez los átomos pueden unirse entre sí a través de enlaces químicos en los cuales intervienen los electrones, estas uniones de átomos son las que forman estructuras más complejas, llamadas “moléculas”.

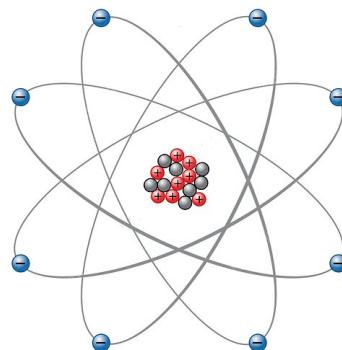


Fig. 1.1: Estructura del átomo de Oxígeno compuesto por 8 neutrones y 8 protones en su núcleo y rodeado por 8 electrones [1].

Formalmente las moléculas son grupos de dos o más átomos unidos entre si mediante enlaces químicos formando una estructura tridimensional en el espacio eléctricamente neutra (a excepción de algunos gases nobles cuyos átomos se consideran moléculas y ciertas sustancias en donde existen enlaces pero no se las considera moléculas, como los metales). También podemos considerar a las moléculas como la parte más ínfima de una sustancia capaz de conservar las propiedades físicas y químicas. Para representarlas existen distintos métodos que varían en su poder expresivo. Una de las formas más simples es la fórmula plana de la molécula que consiste en escribir los símbolos de los átomos que la componen y la cantidad de veces que aparecen presentes, el ácido sulfúrico por ejemplo puede escribirse como H_2SO_4 , indicando la presencia de un átomo de azufre, cuatro de oxígeno y dos de hidrógeno. Este tipo de representación se suele utilizar en los compuestos inorgánicos y en las ecuaciones químicas pero carece de una propiedad importante, no dice nada acerca de los enlaces entre los átomos ni su disposición espacial. En la química orgánica, que estudia las propiedades de moléculas cuyos esqueletos están formados por largas cadenas de carbonos, es muy frecuente la presencia de *isómeros*, moléculas que comparten la misma fórmula molecular pero difieren en su estructura, este tipo de compuestos suele representarse a través de la fórmula estructural, una forma de mostrar la molécula gráficamente indicando las posiciones y los enlaces entre los átomos que la componen.

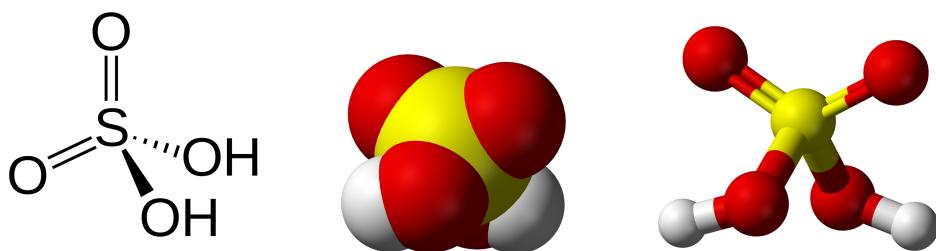


Fig. 1.2: Distintas representaciones del ácido sulfúrico, de izquierda a derecha, fórmula estructural, modelo calotte, modelo de barras y esferas [2][3].

Existe una variedad de estándares disponibles para representar las moléculas estructuralmente, entre ellos se encuentran SMILES, InChI y CML. Este tipo de estándares permiten codificar mediante cadenas de texto muchas propiedades acerca de las moléculas que son fundamentales para describir su comportamiento y su conformación. Estas cualidades incluyen los enlaces atómicos, propiedades estructurales de los isómeros, cargas eléctricas e isótopos, entre otras tantas.

Para construir modelos de aprendizaje automático es necesario abstraer ciertas propiedades de los objetos sobre los cuales los modelos van a realizar predicciones. Para aumentar las probabilidades de obtener modelos precisos, se requiere que la información provista a los algoritmos sea lo más representativa posible, e aquí la necesidad de tener buenas abstracciones de las propiedades de las moléculas. No obstante, modelar exactamente la información molecular tal como se da en la naturaleza resulta casi imposible, ya que estas se encuentran en constante movimiento, estirándose y torsiéndose. Para compensar esto existen formas de modelar la estructura en tres dimensiones que aportan datos sobre los ángulos y distancias de los enlaces, de forma de generar un mapa tridimensional que captura mejor la disposición de los átomos. Este tipo de representación es más compleja de generar y no se

encuentra presente en muchas bases de datos por lo que en este trabajo nos conformamos con obtener la representación bidimensional dada por las fórmulas estructurales.

1.3. El sabor y su relación con el gusto y el olfato

Bases biológicas

Los sabores son sensaciones causadas por sustancias químicas tales como alimentos o medicamentos entre otras y están determinados por la percepción de dichos químicos en el tracto olfativo y gustativo. Al ser transportadas por el aire, las moléculas ingresan al conducto nasal y estimulan los *cilios*, células nerviosas receptoras que se encuentran en la membrana mucosa y poseen extensiones similares a pelos, desencadenando así una serie de impulsos nerviosos que atraviesan el hueso superior de la cavidad nasal(lámina cribosa) y se conectan finalmente a unas células nerviosas llamadas **bulbos olfatorios**, los nervios asociados al olfato. El impulso viaja por estos nervios llegando al cerebro el cual los interpreta como olores, y estimulan además el lóbulo temporal encargado de almacenarlos como recuerdos. Por otro lado, las **papilas gustativas** contienen múltiples receptores del gusto compuestos también por cilios, cada tipo de receptor se asocia a alguno de los cinco sabores básicos: dulce, ácido, amargo, salado y sávido(umami). Si bien se cree que existen zonas de la lengua más sensibles a ciertos sabores, toda la lengua es capaz de percibir todos los sabores. De esta forma los alimentos estimulan los cilios de la lengua y desencadenan al igual que el olfato una serie de impulsos nerviosos conectados a los nervios craneales del gusto, los nervios faciales y glosafaríngeos los cuales transportan la información al cerebro que los interpreta como sabores distintos.

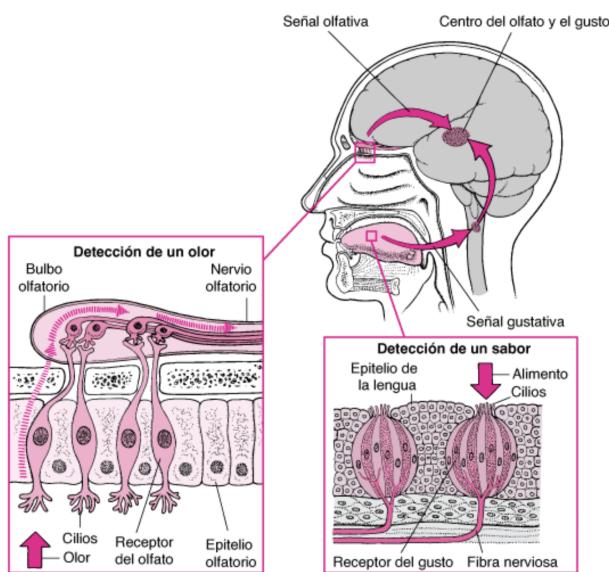


Fig. 1.3: Representación de los sistemas olfativo y gustativo [4].

La combinación de la información sensorial relacionada al olor y al gusto, junto con la textura y temperatura de la comida o sustancia ingerida se procesa de forma única por el cerebro generando los distintos sabores que percibimos. Los mecanismos subyacentes de estos sistemas se encuentran directamente relacionados entre si por lo que cuando hablamos de sabores estamos indistintamente haciendo referencia a ambos sistemas, tanto olfativo como gustativo.

Es necesario destacar que los sabores no son únicamente fenómenos biológicos y que se encuentran también influenciados por otros aspectos como la cultura y la edad. En un estudio[30] investigadores compararon la percepción de distintos olores en grupos de personas compuestos por canadienses y franceses, algunos de estos olores relacionados con la cultura y la historia de cada sociedad como el arce y la lavanda y otros independientes de ellos. Descubrieron que los canadienses presentaban mayor sensibilidad al percibir los olores del arce respecto de los franceses, y por otro lado los franceses resultaron más sensibles al olor de la lavanda. Cuando evaluaron la sensibilidad olfativa con compuestos como la fresa y la rosa, los cuales no presentan un vínculo particular con ninguna de ambas culturas, no se encontraron diferencias destacables. En otros estudios[26][35][36][22] investigadores encontraron que la percepción del olor también se encuentra influenciada por la edad de la persona, por ejemplo los niños (<16) y los adultos mayores (>55) resultaron ser menos sensibles a los olores que los adultos de mediana edad. Estas características particulares de los sabores complejizan aun más su clasificación ya que introducen un sesgo cognitivo inherente al ser humano.

Estructura del sabor

Si nos remitimos a uno de los epítomes de la perfumería clásica podemos detenernos un momento en el primer perfume de la casa Chanel, el famoso Chanel n5 creado por el perfumista Ernst Beaux, lanzado en 1921 y aún presente en la actualidad más de cien años después de su aparición. Como sucedió con muchas obras notables en las artes y las ciencias, una parte del proceso que derivó en su concepción estuvo regido por el azar. Cuando se encontraban preparando distintas muestras perfumísticas para presentar ante Coco Chanel, uno de los estudiantes de laboratorio de Beaux confundió una mezcla pura de aldehídos con otra diluida al diez por ciento y la utilizó en la base de lo que serían las futuras muestras. Esta confusión causó una concentración excesiva de aldehídos nunca antes utilizada en un perfume. Luego Beaux preparó diez viales numerados del 1 al 5 y del 20 al 24 y se los presentó a Chanel la que exclamó con seguridad: “Número cinco. Sí”, dando nacimiento al perfume más emblemático de dicha casa. más tarde Coco proclamó:

“Eso es lo que estaba esperando. Un perfume como ningún otro. Un perfume de mujer, con el aroma de una mujer.”

Coco Chanel, 1921

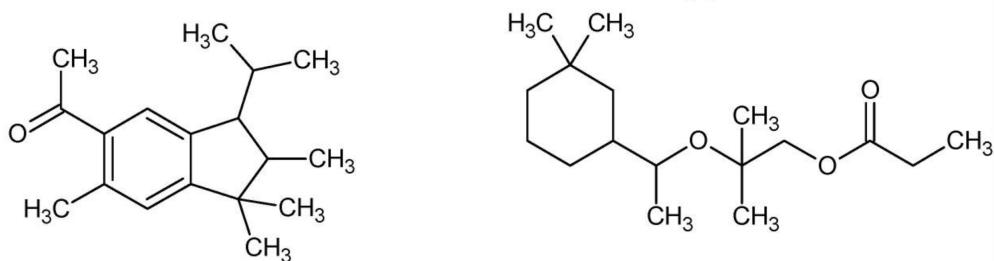
Los aldehídos son los compuestos que dieron a Chanel n5 su característico olor, estos son una clase de moléculas orgánicas que se caracterizan por poseer un grupo funcional carbonilo ($-CHO$) y en la industria perfumística se asocian a olores generalmente placenteros, por ejemplo el “cinamaldehído” ($C_6H_5CHCHCOH$) se asocia al sabor de la canela,

el “geranal” ($C_9H_{15}CHO$) al limón y el “nonanal” ($C_8H_{17}CHO$) a la rosa, entre tantos otros. Este ejemplo permite poner en evidencia algunas de las preguntas que despiertan el interés tanto de la industria como de la ciencia teórica y atanen a la comprensión de las estructuras químicas de los compuestos y la relación que guardan con el efecto sensorial que estos nos evocan, es acaso la cantidad de carbonos o hidrógenos lo que diferencia el olor a limón del geranal respecto del olor a rosa del nonanal? Cómo es posible intensificar o suavizar el olor de estas moléculas? Basta con remover o sacar átomos para lograrlo y, en ese caso, qué átomo o conjunto de ellos es el adecuado? Es posible que Beaux y su equipo de químicos se hayan enfrentado a las mismas dudas al querer transmitir la sensación de los jazmines o del iris, y son las mismas preguntas que siguen inquietando a los científicos de hoy en día.

El futuro de la perfumería está en manos de los químicos... Tendremos que confiar en los químicos para encontrar nuevos productos químicos si queremos hacer acordes nuevos y originales

Ernst Beaux

Muchos estudios se han dedicado a establecer una correspondencia entre la estructura molecular de los compuestos y su olor [28][33][29] y si bien algunos olores pueden ser similares en base al grupo funcional que presentan en la molécula, como puede ser el caso de algunos aldehídos, las aminas que se caracterizan por un olor animálico similar al pescado o los ésteres que se asocian a sabores frutales y florales, esto no funciona como método general y son pocos los casos en los que aplica. Existen múltiples ejemplos estudiados en la literatura en donde estructuras químicas completamente distintas tienen olores similares, esto sucede por ejemplo con el olor del almizcle, como se observa en la siguiente imagen de dos moléculas cuyo olor es similar pero estructuralmente son muy distintas.



1-[1,1,2,6-tetramethyl-3-(propan-2-yl)-2,3-dihydroindene-5-yl]ethanone 2-(1-(3,3-dimethylcyclohexyl)ethoxy)-2-methylpropyl propionate

Fig. 1.4: La estructura de las moléculas es completamente distinta pero su olor evoca el almizcle en ambos casos [5].

Contrariamente al ejemplo anterior, existen moléculas cuya estructura es casi idéntica y su olor difiere, por ejemplo el “limoneno” es una molécula quiral, lo que significa que no es superponible con su imagen espejada (lo mismo que sucede si queremos superponer la mano izquierda con la mano derecha). En su presentación natural más común el (R)-(+)-limoneno, la molécula tiene olor a naranja, pero en su formula (S)-(-)-limoneno, que

consiste en rotar la estructura sobre uno de sus carbonos (llamado “centro quiral”), se obtiene una molécula con olor a limón.

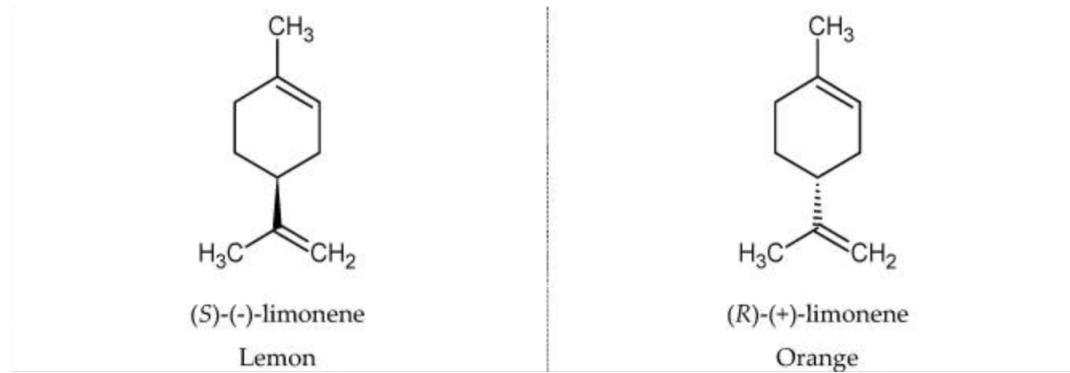


Fig. 1.5: La estructura de la molécula es casi idéntica salvo por una porción de ella que se encuentra rotada en el espacio, lo que confiere olores distintos [5].

Estos ejemplos ponen en evidencia la complejidad subyacente que existe al tratar de comprender las características olfativas y gustativas de las moléculas en base a la conformación de su estructura macroscópica. Semejante dificultad es lo que motiva a utilizar otros enfoques para atacar este problema y es el motivo de este trabajo, en el cual pretendemos extraer patrones complejos de los datos estructurales con la utilización de redes neuronales.

1.4. Grafos

Un *grafo* es una estructura matemática que puede representar relaciones entre un conjunto de elementos. Un grafo se representa como $G = (V, E)$ donde V es un conjunto de nodos o vértices y E es un conjunto de ejes o relaciones entre nodos. Si v_1, v_2 son dos vértices, de haber un eje e_1 que los conecte, se lo denota como $e_1 = (v_1, v_2)$ o (v_2, v_1) . En particular si existe un eje e_k tal que $e_k = (v_i, v_j)$, entonces v_i y v_j se consideran vecinos. Existen dos tipos de grafo, *dirigido* y *no-dirigido* o simplemente, grafo. En el dígrafo los ejes pasan a tener dirección, por lo tanto entre dos vértices v_1 y v_2 pueden existir dos ejes $e_1 = (v_1, v_2)$ y $e_2 = (v_2, v_1)$. Mientras que en el grafo, $e_1 \neq e_2$. Por último, definimos el *grado* de un nodo, como la cantidad de vecinos adyacentes que tiene. En el caso de los dígrafos, se divide el grado en *in_degree* la cantidad de ejes que llegan a un nodo y *out_degree* los ejes que salen de un nodo. Definimos un *camino* en un grafo como una secuencia de nodos vecinos. Decimos que dos vértices están *conectados* si existe un camino de uno al otro. En particular se llama camino simple, al camino del grafo que no repite nodos en su secuencia. Luego definimos *ciclo* al camino simple que empieza y termina en el mismo nodo. Por otro lado, se puede distinguir la conectividad de un grafo. Se dice que el grafo es *conexo* si existe un camino entre todo par de nodos. Podemos hablar de grafos más (o menos) conexos según la cantidad de caminos entre dos pares de vértices cualquiera.

Un grafo puede utilizarse para representar por ejemplo una red de computadoras, una red de sitios web conectados entre sí a través de hipervínculos y relaciones de amistad o seguimiento en una red social.

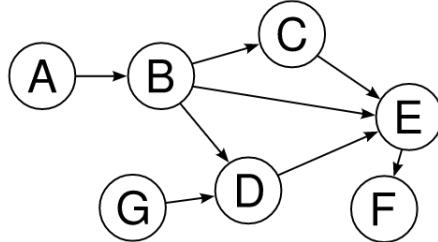


Fig. 1.6: Ejemplo de un grafo dirigido

Para implementar algoritmos sobre grafos, debemos poder representarlos en la computadora. Los dos enfoques más comunes suelen ser a través de matrices de adyacencia y listas de adyacencia.

Matriz de adyacencia

Consiste en representar el grafo a través de una matriz cuadrada cuya dimensión es igual a la cantidad de nodos del grafo. Dado un grafo G , se define su matriz de adyacencia $A \in \mathcal{R}^{n \times n}$, $A = [a_{ij}]$ como:

$$a_{ij} = \begin{cases} 1 & \text{si } G \text{ tiene una arista entre } v_i \text{ y } v_j \\ 0 & \text{si no} \end{cases}$$

Su principal ventaja radica en que al ser un array el tiempo de acceso es constante y por lo tanto también lo es saber si existe un eje entre dos nodos, como contra parte el uso de memoria no es óptimo, ya que independientemente de que el grafo contenga pocos o muchos ejes el espacio utilizado es siempre el mismo.

Lista de adyacencia

Consiste en representar el grafo utilizando una lista de pares de ejes. Contiene solo los ejes pertenecientes al grafo y por lo tanto el uso de memoria es óptimo.

Dado un grafo G , se define su lista de adyacencia como l tal que $(v_i, v_j) \in l$ si y solo si G tiene una arista entre v_i y v_j . Su principal desventaja radica en que para saber si un eje pertenece a G se debe recorrer toda la lista en el peor de los casos obteniendo una complejidad lineal en la cantidad de ejes $O(|E|)$.

Las moléculas como grafos

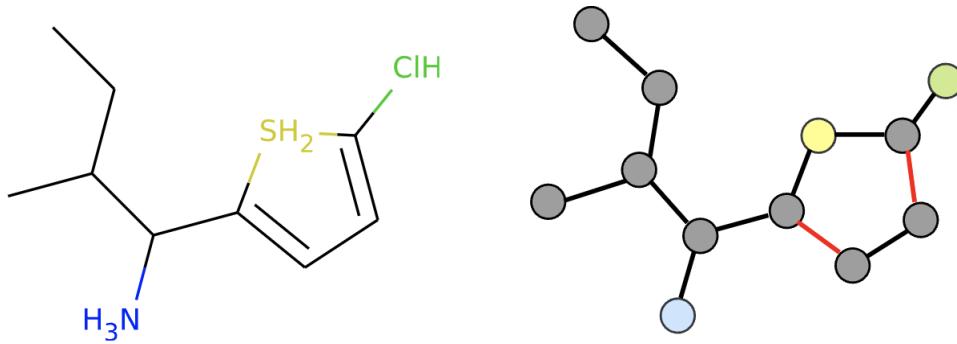
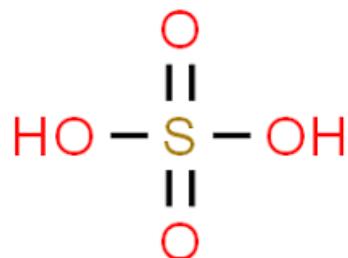


Fig. 1.7: Representación de una molécula mediante un grafo [6].

Respecto a las moléculas, recordemos que se componen de conjuntos de átomos unidos entre si mediante enlaces químicos. En este sentido podemos establecer una correspondencia entre este concepto y la estructura de un grafo conexo no dirigido, conexo porque no pueden haber átomos separados de la estructura y no dirigido porque los enlaces se dan de forma simétrica entre ambos átomos y no de uno hacia el otro en una sola dirección. Por un lado el conjunto de nodos V va a estar dado por los átomos que componen la molécula mientras que el conjunto de ejes E va a estar dado por todos los pares de nodos (v_i, v_j) tales que los átomos representados por v_i y v_j se encuentran unidos en la molécula mediante un enlace. De esta forma podemos representar la estructura de la molécula matemáticamente, por otro lado, existen muchas propiedades asociadas a la molécula que se relacionan con los átomos y los enlaces, este tipo de información suele representarse utilizando matrices adicionales. Si queremos por ejemplo codificar tres atributos sobre los átomos como el símbolo, la carga y la masa, una matriz con n filas, siendo n la cantidad de átomos de la molécula, y tres columnas puede albergar esta información y así es posible generar matrices de tamaño arbitrario capaces de contener todos los atributos que se requieran modelar.

Tomemos nuevamente el ejemplo de la molécula del ácido sulfúrico H_2SO_4 :

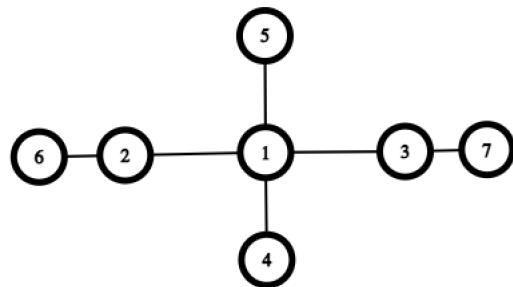


Esta se compone de siete átomos unidos mediante distintos enlaces. Para obtener su representación como grafo debemos primero asignar los átomos a nodos y luego los enlaces a ejes del grafo. Como la molécula consta de siete átomos el grafo debe contener exactamente siete nodos, por lo que el conjunto de vértices estará dado por $V = \{1, 2, 3, 4, 5,$

$\{6, 7\}$. Un posible mapeo entre los átomos y los nodos podría ser el siguiente:

$$\begin{aligned} S &\longrightarrow 1 \\ O &\longrightarrow 2 \\ O &\longrightarrow 3 \\ O &\longrightarrow 4 \\ O &\longrightarrow 5 \\ H &\longrightarrow 6 \\ H &\longrightarrow 7 \end{aligned}$$

Este mapeo condiciona el conjunto de ejes, el cual debe ser consistente con los enlaces en la molécula, de la imagen anterior sabemos que el átomo de azufre se encuentra unido a los cuatro átomos de oxígeno y hay dos de ellos que están enlazados a un átomo de hidrógeno. Basándonos en el mapeo de átomos-nodos que dimos el conjunto de ejes quedaría conformado por $E = \{(1, 2), (1, 3), (1, 4), (1, 5), (2, 6), (3, 7)\}$. El grafo en cuestión puede ser representado gráficamente de la siguiente forma:



Por otro lado, dado que el grafo esta codificando solamente información estructural de la molécula, para codificar los atributos de los ejes y los átomos se deben utilizar matrices distintas. Supongamos que se quieren representar tres atributos de los átomos, el símbolo, la masa y la carga formal, y un atributo de los ejes que contenga el tipo de enlace (doble, triple, simple, polar, etc). El primer conjunto puede representarse con una matriz $F_n \in \mathcal{R}^{7 \times 3}$ y el segundo en una matriz $F_e \in \mathcal{R}^{6 \times 1}$, dado que hay siete átomos y tres atributos por átomo en la primera y seis ejes y un atributo por eje en la segunda. Denotando s_i, c_i, m_i a los símbolos, cargas y masa del i -ésimo átomo y t_j el tipo del j -ésimo enlace, las matrices de atributos y la matriz de adyacencia del grafo quedarían denotas por:

$$F_n = \begin{bmatrix} s_1 & c_1 & m_1 \\ s_2 & c_2 & m_2 \\ s_3 & c_3 & m_3 \\ s_4 & c_4 & m_4 \\ s_5 & c_5 & m_5 \\ s_6 & c_6 & m_6 \\ s_7 & c_7 & m_7 \end{bmatrix} \quad F_e = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \end{bmatrix} \quad Adj = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Esta representación matricial es la que permite definir los operadores convolucionales sobre el grafo y es la que usaremos para codificar los atributos del grafo. El ejemplo anterior

representa una ilustración simplificada que no contempla todos los posibles atributos de los enlaces y de los átomos pero sirve para comprender como se estructura dicha información.

1.5. Redes Neuronales Artificiales

Las redes neuronales artificiales (RNA) son un subconjunto de algoritmos y técnicas de aprendizaje dentro del mundo del machine learning que han ganado popularidad en las ultimas décadas gracias al desarrollo científico y al aumento exponencial de información y bases de datos disponibles para su implementación, particularmente las RNA son la base de lo que se conoce como aprendizaje profundo. Su origen se remonta al año 1943 cuando McCulloch y Pitts presentaron el primer modelo matemático de una neurona al que hoy denominamos **perceptrón** y es la unidad fundamental sobre la que se construyen las RNA. Como su nombre lo indica, estas redes guardan una estrecha relación con las redes neuronales que componen el sistema nervioso al intentar modelar el proceso en que la información se propaga a través de las neuronas. Resumidamente, las neuronas son células que se comunican entre si mediante impulsos electroquímicos al recibir estímulos de otros conjuntos de neuronas y, en base a ciertas propiedades específicas de la neurona, pueden replicar el impulso recibido hacia las neuronas siguientes o cortar con el flujo de estímulos reduciendo la transducción de señales, a este proceso se le llama **sinapsis** y es lo que modelan las neuronas artificiales.

Perceptrón

Dado un vector de entrada $\langle x_1, \dots, x_m \rangle = X \in \mathcal{R}^m$, definimos la salida de un perceptrón como una función dada por la siguiente expresión:

$$\hat{y} = g\left(b + \sum_{i=1}^m x_i w_i\right)$$

Donde $\langle w_1, \dots, w_m \rangle = W \in \mathcal{R}^m$ es un vector de **pesos**, $b \in \mathcal{R}$ es un valor de umbral, también llamado **bias** y g es una función no lineal llamada **función de activación**.

La idea detrás de este concepto es modelar la sinapsis neuronal, para esto el perceptrón debe ser configurado con un vector de pesos W y un bias b de modo que para ciertos vectores de entrada X , que pueden ser pensados como estímulos o señales, retorne valores altos, emulando la propagación del estímulo, o bien reduzca la potencia del estímulo al retornar valores cercanos a cero. La función de activación se utiliza porque sin ella, el input y el output estarían condicionados por una relación lineal y en muchos problemas es requisito que el modelo sea capaz de generar representaciones de los datos arbitrarias cuya relación no es lineal. Finalmente tanto los pesos del perceptrón como el umbral deben ser ajustados a través de algún algoritmo de aprendizaje de modo que se ajusten al problema y los datos.

La RNA más simple que podemos crear es formada por un solo perceptrón, pero también es posible agrupar conjuntos de perceptrones para formar **capas**. Las capas pueden considerarse conjuntos de neuronas que funcionan a un mismo nivel, ya que todos sus perceptrones reciben el mismo vector de información y propagan su salida hacia la capa

siguiente. A su vez, las capas pueden apilarse sucesivamente una tras otra para crear arquitecturas más profundas, y es de ahí de donde proviene el término *deep* en deep learning. Al agregar capas a la red el modelo se torna más complejo y la cantidad de parámetros configurables aumenta, permitiéndole a la red hallar patrones más complejos en la información, sin embargo esto suele requerir de grandes cantidades de datos para que el modelo sea capaz de generalizar a nuevos ejemplos nunca antes vistos.

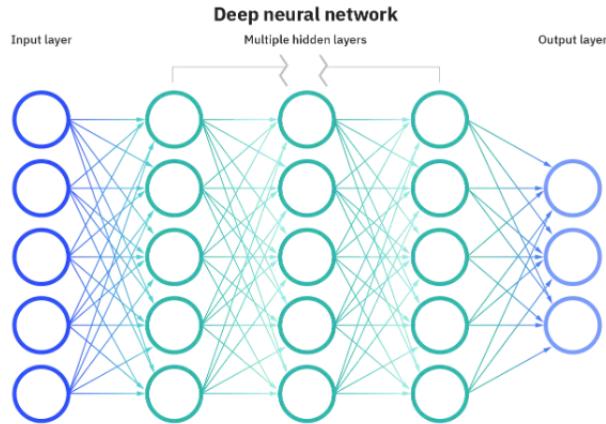


Fig. 1.8: En esta imagen vemos una red formada por cinco capas, a las tres capas intermedias se las llama capas ocultas, ya que no son ni la capa de entrada ni la capa de salida. Cada capa se compone de un conjunto de perceptrones, denotados por los círculos de color. La salida de todos los perceptrones forma un nuevo vector que alimenta los perceptrones de la capa siguiente [7].

En estas redes la información fluye de atrás hacia adelante sin formar ciclos entre la salida de los perceptrones de una capa y las capas anteriores, es por esto que se las suele llamar **feedforward networks** o simplemente perceptrones multicapa (MLP).

Para que las RNA puedan aprender de la información y generalizar correctamente es necesario entrenarlas con ejemplos y penalizarlas cuando realizan una predicción incorrecta ajustando los parámetros del modelo, para esto es necesario definir una **función de pérdida** que indique la magnitud del error cometido por la red en base al valor de salida y al valor esperado para los ejemplos utilizados. El error empírico suele calcularse mediante la siguiente expresión en base a la función de perdida:

$$\mathcal{J}(W) = \frac{1}{n} \sum_{i=1}^n \mathcal{L}(f(x^{(i)}; W), y^{(i)})$$

Donde \mathcal{J} representa el error medio cometido por la red luego de procesar n ejemplos con el conjunto de parámetros W . El output predecido por la red para cada ejemplo está dado por la función $f(x^{(i)}; W)$ y la salida esperada por $y^{(i)}$, mientras que \mathcal{L} es la función de perdida encargada de cuantificar la diferencia entre la salida de la red y la salida esperada, algunas de estas funciones utilizadas ampliamente en la literatura son la **entropía**

cruzada y el error cuadrático medio, entre tantas otras.

La idea detrás del entrenamiento consiste en encontrar el conjunto de parámetros \mathbf{W}^* que minimice $\mathcal{J}(W)$:

$$\mathbf{W}^* = \operatorname{argmin}_W \frac{1}{n} \sum_{i=1}^n \mathcal{L}(f(x^{(i)}; W), y^{(i)})$$

$$\mathbf{W}^* = \operatorname{argmin}_W \mathcal{J}(W)$$

Este proceso es llamado **optimización** y consiste en ir haciendo pequeñas modificaciones de los parámetros W iterativamente hasta converger a un error aceptable. El método utilizado por las redes neuronales y muchos otros modelos de aprendizaje automático se llama **descenso por gradiente**. El algoritmo de descenso por gradiente comienza inicializando aleatoriamente los pesos de la red siguiendo alguna distribución estadística, luego procede a computar el gradiente del error promedio en función de W con los ejemplos del dataset y ajusta los parámetros de la red desplazándolos en la dirección contraria del gradiente de modo que se avance hacia un mínimo de la función. En este ultimo paso el gradiente es reescalado por un factor η denominado **learning rate** que determina la magnitud del cambio. Estos dos últimos pasos que consisten en calcular el gradiente y ajustar los parámetros se realizan iterativamente hasta converger a un error aceptable. Podemos sintetizar el algoritmo general de entrenamiento para un red neuronal mediante el siguiente pseudocódigo:

```

1:  $W \leftarrow$  inicializar pesos del modelo con distribución  $\mathcal{N}(0, \sigma^2)$ 
2: while no se alcanzó el criterio de convergencia do
3:    $\nabla \mathcal{J}(W) \leftarrow$  calcular gradiente de  $\mathcal{J}$  en función de  $W$ 
4:    $W \leftarrow W - \eta \nabla \mathcal{J}(W)$ 
5: end
6: retornar matriz de pesos  $W$ 
```

El método para calcular los gradientes utiliza la regla de la cadena del cálculo para obtener las derivadas parciales de \mathcal{J} en función de los parámetros, a este método se lo conoce en la literatura como **backpropagation**. Una vez que se alcanza el criterio de convergencia para el error, el algoritmo se detiene y retorna los pesos de la red que definen al modelo.

Las redes aquí presentadas constituyen la base de las redes neuronales artificiales, en la actualidad existen modelos más complejos especialmente diseñados para abordar distintos tipos de problemas y modelos de datos, ejemplo de esto son las **redes convolucionales** que han ganado gran importancia en el campo de detección de objetos y clasificación de imágenes y las **redes recurrentes** en el procesamiento del lenguaje, en esta tesis implementamos y estudiamos las particularidades de las **redes de grafos convolucionales**, una familia de modelos que guarda cierta semejanza con las redes convolucionales y están siendo aplicadas en el modelado de grafos con muy buenos resultados.

2. OBTENCIÓN Y ESTRUCTURACIÓN DE LOS DATOS

2.1. Flavor DB

FlavorDB es un repositorio que se compone de 25.595 moléculas que se encuentran etiquetadas con sus respectivos sabores y olores. Entre ellas, 2.254 moléculas están asociadas a 936 ingredientes naturales pertenecientes a 34 categorías. FlavorDB contiene una interfaz dinámica y fácil de utilizar la cual facilita la exploración de moléculas de sabor para encontrar aquellas que coincidan con un sabor o estructura deseados; explorar moléculas de un ingrediente; descubrir nuevas combinaciones de alimentos; encontrar la esencia molecular de los ingredientes alimentarios; asociar características químicas con un sabor y más. Dentro del repositorio cada molécula contiene los datos asociados a su nombre común, su nombre IUPAC (la International Union of Pure and Applied Chemistry es una organización internacional que rige como autoridad reconocida en el desarrollo de estándares para denominación de compuestos químicos), los grupos funcionales que contiene, sus representaciones en InChI, un estándar utilizado para identificar y describir compuestos químicos que describiremos en detalle en la próxima sección, y sus perfiles de sabor y olor con las etiquetas correspondientes. En la siguiente imagen vemos un ejemplo de la molécula *Vanilina* obtenido de la página de FlavorDB.

Vanillin

Molecular & Flavor Profile	
Common name:	Vanillin
IUPAC name:	4-Hydroxy-3-Methoxybenzaldehyde
SMILES:	<chem>COC1=C(C=CC(=C1)C=O)O</chem>
CAS:	121-33-5, 121-33-5, 84650-63-5
Flavor Profile:	Vanilla, Chocolate, Sweet, Creamy
FEMA Flavor Profile:	Vanilla
FEMA Number:	3107
Taste:	pleasant vanilla taste
Odor:	sweetish smell pleasant aromatic vanilla odor
Functional Groups:	Carbonyl Compound, Aldehyde, Hydroxy Compound, Phenol Or Hydroxyhetarene, Ether, Alkyl Aryl Ether, Aromatic Compound



The chemical structure of vanillin is shown as a benzene ring substituted with a hydroxyl group (-OH) at position 4 and a methoxy group (-OCH₃) at position 3. A formyl group (-CHO) is attached to the ring at position 1.

[View JSmol](#)

[Search Similar in FlavorDB](#)

[ZINC Similarity Search](#) 

Fig. 2.1: Registro de la molécula Vanilina en FlavorDB [8].

Para armar el dataset utilizamos la api de FlavorDB que permite obtener los datos de cada molécula en formato *json* indexando cada una por su pubchem id. Luego procedimos a descargar los datos de todas las moléculas y compilamos un archivo en formato csv conteniendo en cada fila los datos de las moléculas con sus respectivas representaciones en InChI y sus perfiles de sabores y olores, estos últimos fueron utilizados para realizar la segmentación entre distintas categorías en los experimentos con criterios específicos según el objetivo final de cada uno.

2.2. Explorando los tags

En la imagen previa podemos observar un campo en particular, **Flavor Profile**, el cual contiene las etiquetas con sabores y olores a los que se asocia la molécula dada y es particularmente el que utilizamos para generar la segmentación de los datos, en el caso de la *Vanilina* esta se asocia a las etiquetas “vanilla”, “chocolate”, “sweet” y “creamy”, por lo tanto dicha molécula queda directamente asociada con estas categorías. Esto significa que si queremos entrenar un modelo capaz de distinguir entre moléculas con sabor a chocolate y moléculas sin sabor a chocolate, la Vanilina va a pertenecer al primer conjunto dado que contiene el etiqueta “chocolate”.

Luego de compilar el dataset procedimos a analizar la distribución de las distintas etiquetas presentes en las moléculas de forma de entender que datos teníamos presentes para utilizar y en qué proporciones. El repositorio de FlavorDB particularmente contiene una proporción muy superior de moléculas cuya única etiqueta asociada es “sweet” o “sweet-like”, 8.171 y 13.782 moléculas respectivamente, probablemente debido al hecho de que existe mucho interés en el descubrimiento de endulzantes capaces de reemplazar el azúcar, sustancia con muchos impactos nocivos en la salud.

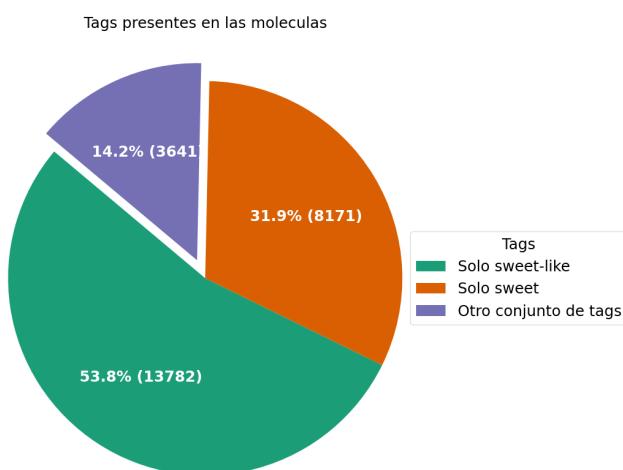


Fig. 2.2: Cerca del 85 % de las moléculas contienen una sola etiqueta perteneciente a la clase “sweet” o “sweet-like”, mientras que el restante son moléculas con otras combinaciones de etiquetas.

Debido a la prevalencia de las etiquetas *sweet* y *sweet-like* y al hecho de que en su mayoría aparecen sin otras etiquetas asociadas, optamos por establecer un criterio distinto para la categoría “*sweet*”. En este criterio consideramos la etiqueta válida solo para aquellas moléculas que además de contener al tag “*sweet*” contuvieran alguna otra etiqueta asociada, como podrían ser “*creamy*” o “*chocolate*”. Un ejemplo de esto es la molécula Vanilina, la cual contiene cuatro etiquetas y una de ellas es “*sweet*”, para esta molécula sí contamos como válida la etiqueta.

En todo problema que desee ser abordado con algoritmos de aprendizaje automático resulta indispensable comprender la naturaleza y la distribución de la información con que se trabaja, tener una gran abundancia de datos para entrenar los modelos suele ser lo ideal, pero no siempre resulta posible. Dado que nuestros experimentos se centraron en distinguir y predecir etiquetas en las moléculas, necesitamos que las etiquetas utilizadas sean lo más frecuentes posible, pues si una etiqueta solo aparece en una pequeña cantidad de moléculas, estas no serán suficientes para que el modelo pueda aprender de los ejemplos. A continuación vemos como se distribuye la frecuencia de apariciones de las etiquetas del dataset y en los percentiles más frecuentes.

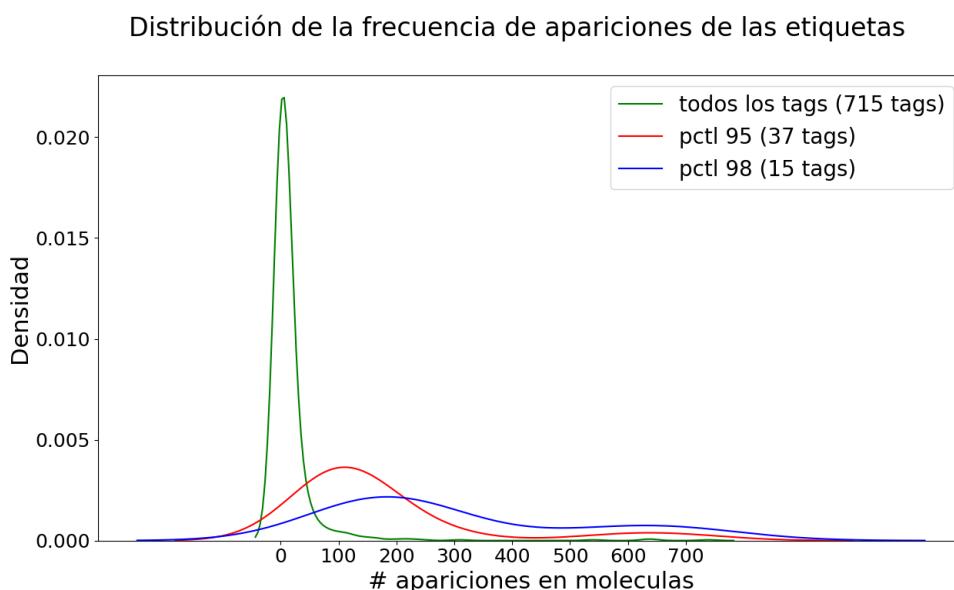


Fig. 2.3: La frecuencia de aparición de todas las etiquetas (verde) resulta en una distribución cuya media ronda el valor 4, esto significa que en promedio cada etiqueta aparece en cuatro moléculas solamente, indicándonos que la gran mayoría de las etiquetas no va a servir como criterio de clasificación. Si miramos el percentil 95 (rojo), las 37 etiquetas promedian alrededor de 120 apariciones mientras que dentro del percentil 98 (azul) rondan las 200.

Dado que para entrenar los modelos necesitamos armar conjuntos de validación, testeo y entrenamiento, las etiquetas a considerar para la experimentación debían ser aquellas presentes en un número de moléculas considerable. Debido a esto se tuvieron en cuenta

solamente las etiquetas pertenecientes al percentil más alto (98) para los modelos de sabores. En la siguiente imagen podemos observar cuáles son estas etiquetas y en cuantas moléculas aparece presente cada una.

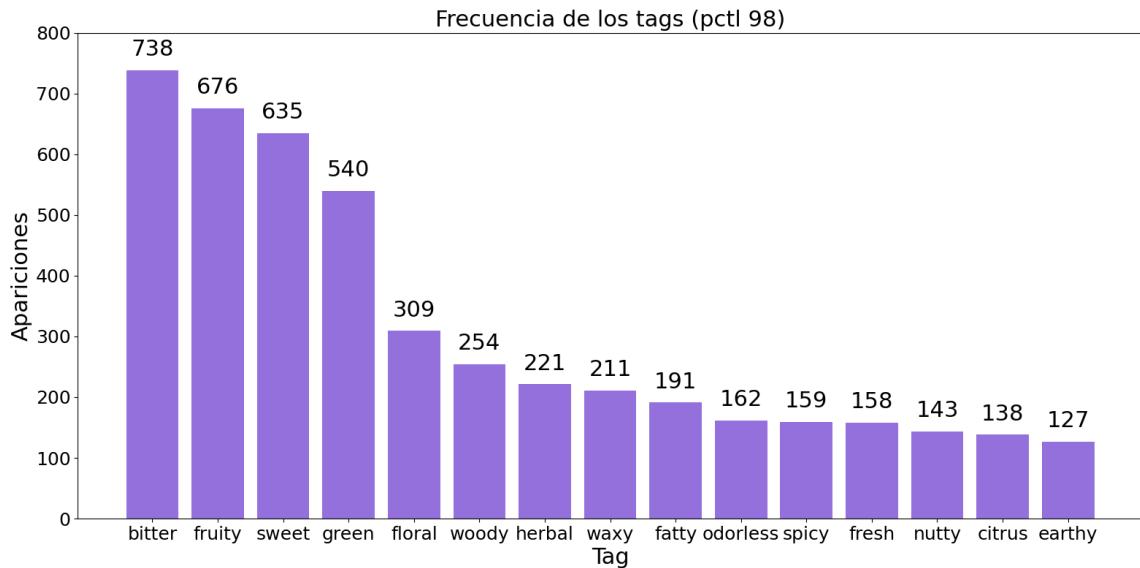


Fig. 2.4: El histograma muestra la cantidad de ejemplos de entrenamiento disponibles en el dataset para cada una de las quince etiquetas más frecuentes (pctl 98).

Finalmente, luego de explorar el dataset y definir los criterios para seleccionar las posibles categorías de clasificación, obtuvimos un conjunto de etiquetas y moléculas significativamente más chico respecto del original y si bien en su mayoría las etiquetas más frecuentes no son tan numerosas como podría desearse, en términos generales consideramos que eran suficientes para armar datasets de entrenamiento razonables. Dependiendo de los experimentos y clasificadores realizados se tomaron distintos subconjuntos de estas etiquetas que se darán a conocer en detalle en cada caso particular.

3. PREPROCESAMIENTO, DE INCHIS A GRAFOS

3.1. InChis

Para entrenar modelos capaces de distinguir propiedades moleculares, primeros debemos hallar una representación digital que permita codificar la información de las moléculas y sirva como input de los algoritmos de aprendizaje automático. Particularmente, como utilizamos redes de grafos convolucionales diseñadas para trabajar con grafos, fue necesario establecer una correspondencia entre la representación que se tenía de las moléculas en el dataset de FlavorDB y los grafos, de modo tal que sea posible implementar algoritmos que lleven a cabo la transformación.

Tomemos como ejemplo la molécula del etanol, cuya formula química es C_2H_5OH . En la siguiente imagen vemos la disposición de los átomos representada en dos dimensiones:

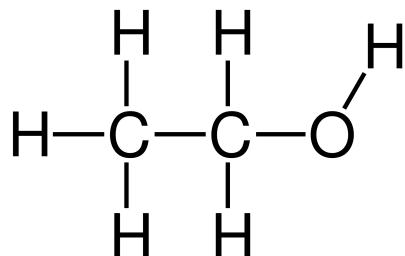


Fig. 3.1: Estructura del etanol.

Ahora bien, si observamos solamente la fórmula de la molécula, C_2H_5OH , esta no nos dice exactamente que átomos se encuentran unidos entre si, simplemente nos indica en qué cantidades estos están presentes, dos átomos de carbono, uno de oxígeno y seis hidrógenos. Si le pedimos a alguien sin conocimientos en química que dibuje la molécula solo con dicha información, hay una variedad de distintas combinaciones que podrían obtenerse para disponer los átomos, por ejemplo podríamos tener el átomo de oxígeno en el centro de los carbonos. Es por esto que solamente la fórmula de la molécula no es suficiente para describir la disposición de sus átomos, menos aún otros aspectos que veremos más adelante como el tipo de los enlaces o características electrónicas. Como mencionamos en la introducción, la información de las moléculas del dataset se encuentra codificada en un estándar llamado InChI, el cual se caracteriza por poseer un alto poder expresivo para representar las propiedades químicas y estructurales.

InChI[25] es el acrónimo de *International Chemical Identifier*, un identificador estándar de sustancias químicas utilizado mundialmente para representar estructuras moleculares mediante cadenas de texto. Estos identificadores describen sustancias químicas en términos de capas de información: los átomos y la conectividad de los enlaces, la información tautómera(asociada a los grupos funcionales), información de isótopos, estereoquímica e información de las cargas electrónicas. La ventaja de utilizar este estándar es que permite

codificar mucha información de la molécula en una simple cadena de texto.

Una de las características centrales de esta codificación es que permite segmentar la información en capas y subcapas, de modo que existe una primera capa llamada “principal” la cual contiene una primera subcapa de información asociada a la fórmula de la molécula indicando la cantidad de apariciones de cada átomo, una segunda subcapa indicando la disposición espacial de los átomos, y una tercera subcapa indicando los enlaces de hidrógeno. Luego existe una variedad de capas adicionales que contienen niveles más específicos de información y pueden ser añadidas de forma modular, entre ellas:

1. Capa de cargas eléctricas
2. Capa estereoquímica
3. Capa isotópica
4. Capa de hidrógenos fijos
5. Capa de reconexiones

La molécula de etanol, cuya fórmula estaba dada por C_2H_5OH , se representa en InChI con la siguiente secuencia de caracteres:

$$\text{InChI} = 1S/C2H6O/c1 - 2 - 3/h3H, 2H2, 1H3$$

En este estándar las capas se separan con el identificador / seguido en algunos casos por una letra que indica de que tipo de capa se trata, en el caso del etanol tenemos cuatro capas:

- $1S$ denota el formato estándar de InChI
- $C2H6O$ indica la numeración y cantidad de los átomos (los seis átomos de hidrógeno no son tenidos en cuenta en la numeración por lo que los átomos 1 y 2 serán carbonos y el tercer átomo será el oxígeno).
- $c1 - 2 - 3$ especifica los enlaces entre átomos, la c inicial sirve para especificar que la capa contiene la información de las conexiones, y $1 - 2 - 3$ denota la estructura carbono-carbono-oxígeno en base a los índices de los átomos.
- por último $h3H, 2H2, 1H3$ denota los enlaces de hidrógeno para cada átomo, esta capa se identifica con la h inicial, luego los índices antes de las hachas mayúsculas denotan el átomo correspondiente y el número que la sigue la cantidad de hidrógenos enlazados al átomo.

Si fuera necesario, como mencionamos previamente es posible añadir más capas con información, en la siguiente imagen tomada de la documentación del estándar de InChI podemos apreciar algunos de los distintos niveles de profundidad que se pueden alcanzar en ciertas moléculas:

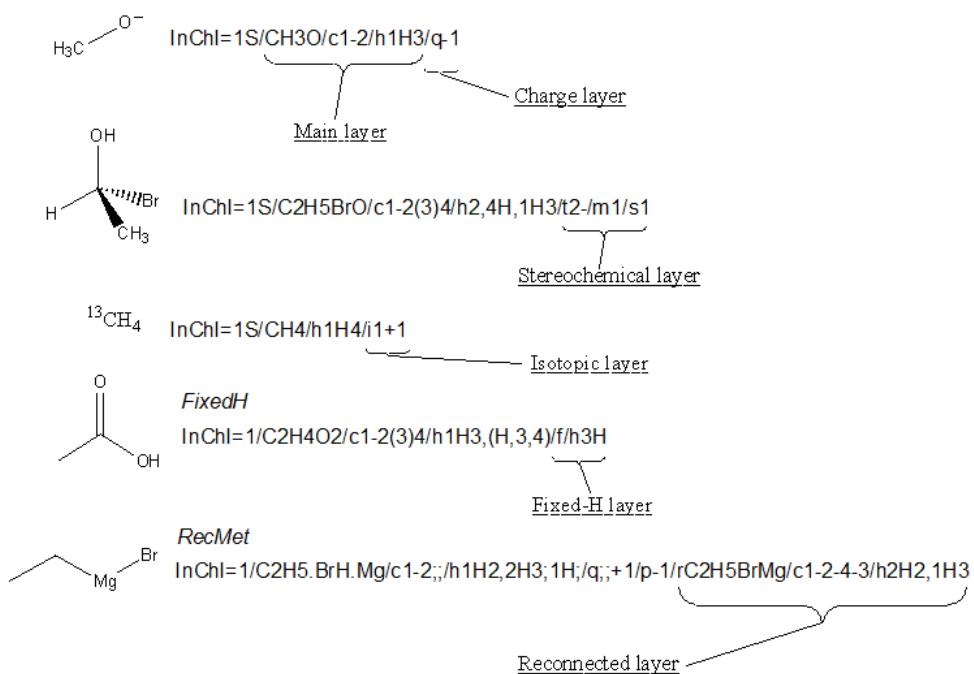


Fig. 3.2: Imagen tomada de la documentación de InChi que ilustra la disposición y organización de las distintas capas [9].

Por último, cabe destacar que existen muchos aspectos y propiedades de las moléculas que no pueden ser capturados en su totalidad por este estándar [25], entre ellos las coordenadas espaciales de los átomos necesarias para representar apropiadamente la molécula en tres dimensiones. No obstante estas limitaciones, InChi es un estándar de facto utilizado ampliamente para representar moléculas y es capaz de codificar un amplio espectro de propiedades deseables de las moléculas.

3.2. RdKit

En el apartado anterior desenmascaramos ciertos aspectos inherentes a la codificación de información molecular y mostramos las propiedades más importantes del estándar utilizado para representar las moléculas, pero, a fin de cuentas, InChi termina con la representación textual de estas. Luego, para trabajar con dicha codificación se necesita de una capa intermedia de software capaz de procesar los InChi's y convertirlos en “moléculas digitales” de modo que podamos extraer la información sin necesidad de parsear la codificación manualmente.

RdKit es una colección de software de quimioinformática y aprendizaje automático escrito en C++ y Python, entre sus módulos se halla *Chem*, el cual brinda funcionalidades especiales para la conversión de InChi a objetos de Python que representan las moléculas y gozan de una interfaz que nos permite acceder cómodamente a todos los datos almacenados en los InChi's.

El siguiente código muestra la sencillez con la que podemos pasar de InChI a moléculas y obtener sus atributos:

Algoritmo 3.1: Extracción de features

```
from rdkit import Chem

ethanol_inchi = "InChI=1S/C2H6O/c1-2-3/h3H,2H2,1H" # representacion como
# string
ethanol = Chem.rdinchi.InchiToMol(ethanol_inchi) # pasaje a objeto

atoms = ethanol.GetAtoms() # devuelve la lista de atomos con objetos de
# tipo Atom
bonds = ethanol.GetBonds() # devuelve lista de ejes con objetos de tipo
# Bond

bond = bonds[0] # obtenemos el primer enlace
type = bond.GetBondType() # SINGLE, DOUBLE, TRIPLE, AROMATIC
is_conjugated = bond.GetIsConjugated() # True, False

atom = atoms[0]
symbol = atom.GetSymbol() # C, O, H, etc
charge = atom.GetFormalCharge() # -2, -1, 0, 1, 2, 3, 4
```

En este código vemos como generar la representación digital de la molécula en *Chem* a partir de su InChI y luego algunos de los mensajes que la interfaz del objeto reconoce como por ejemplo retornar la lista de enlaces y átomos y las propiedades de estos como pueden ser el tipo del enlace y el símbolo del átomo en cuestión. El resto del proceso consiste en obtener las propiedades requeridas de las moléculas dadas por sus enlaces y átomos y generar conjunto de features compatible con los algoritmos de aprendizaje automático.

3.3. Features

Los features son los atributos que se desean modelar en un problema de machine learning y son uno de los aspectos fundamentales de cualquier modelo ya que es la información que estos reciben tanto en el entrenamiento como en producción, por lo tanto deben representar aspectos sustanciales que permitan al modelo establecer relaciones con la información obtenida. Los modelos que utilizamos se caracterizan por realizar convoluciones sobre grafos, esto significa que el modelo espera como input grafos, los cuales contienen embebida en su estructura la información de la molécula. Como mencionamos en la introducción, podemos representar las moléculas como grafos conexos no dirigidos en donde cada átomo es un vértice dentro del grafo y los enlaces entre estos van a representarse por ejes que unen los vértices de dicho grafo. Por otro lado los vértices y ejes van a contener los atributos asociados a los átomos y enlaces de la molécula. A continuación presentamos los atributos que fueron considerados tanto para los átomos como para los enlaces de la molécula y sus respectiva significancia, tales son utilizados por defecto en la literatura y en diversas aplicaciones de aprendizaje automático con moléculas [40].

Atributos de los Enlaces

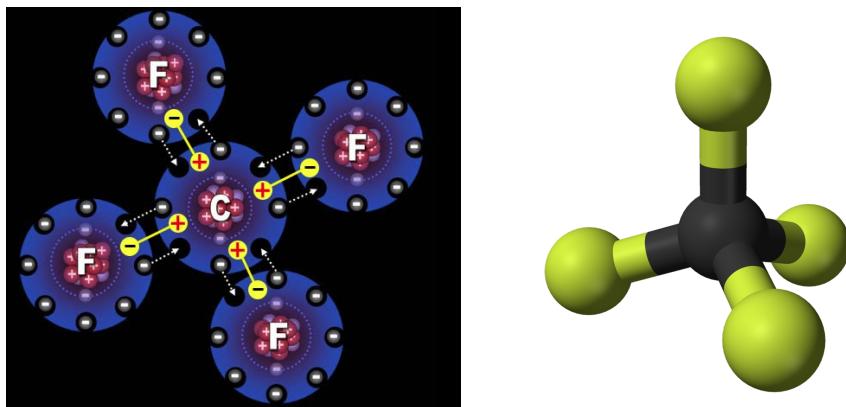


Fig. 3.3: El Tetrafluoruro de Carbono (CF_4), a izquierda sus cuatro enlaces covalentes polares y a derecha su representación tridimensional [10][11].

Para formar moléculas, los átomos deben mantenerse unidos entre si, a dichas uniones se las llama enlaces y son las responsables de mantener la estructura y estabilidad de la molécula. Los enlaces son fundamentales ya que influyen en las propiedades geométricas y químicas de la molécula, por esta razón son parte esencial del modelado. En el plano abstracto, los enlaces están representados por los ejes del grafo y cuentan con las siguientes propiedades:

- **Tipo:** Así como la materia macroscópicamente se distribuye buscando el equilibrio, los gases y los líquidos por ejemplo al expandirse homogéneamente, los átomos se rigen por leyes similares y se unen o separan entre si buscando un equilibrio. Ese equilibrio en los átomos suele darse al completar su orbital más externo con ocho electrones (regla del octeto), y para hacerlo posible comparten e intercambian los electrones de estos orbitales entre si. El cloruro de sodio ($NaCl$) por ejemplo, se forma mediante un enlace *iónico* cuando el sodio (Na) cede el electrón sobrante de su último orbital al cloro (Cl) que le falta un electrón para completar su orbital, formando así la conocida sal de mesa. Existen distintos tipos de enlaces que varían según como se rigen dichos intercambios, los enlaces *covalentes* se dan cuando átomos no metálicos comparten uno o más electrones de su ultimo nivel, si comparten un electrón se llama enlace simple, si comparten dos o tres de los llaman enlaces dobles y triples respectivamente. Los enlaces *aromáticos* por otro lado se dan en estructuras cíclicas como el benceno, un compuesto formado por un anillo de seis carbonos, donde los electrones se comparten entre todos los átomos del enlace y son libres de moverse dentro del anillo. En los enlaces iónicos en cambio uno de los átomos pierde un electrón mientras el otro lo gana, lo que confiere polaridad a la molécula puesto que el átomo que cede su electrón, el catión, pasa a tener carga positiva mientras que el átomo que recibe el electrón, el anión, pasa a tener carga negativa. Existen otros tipos de enlaces, *metálicos*, *flexionados*, etc. Los posibles valores que toman los enlaces de las moléculas en nuestro dataset son enlaces covalentes simples, dobles o triples y enlaces aromáticos.
- **Conjugado:** Los enlaces conjugados se componen de alternancias entre enlaces co-

valentes simples y múltiples, por ejemplo el anillo de benceno cuyos seis carbonos están unidos por enlaces simples y dobles. Posibles valores: True, False

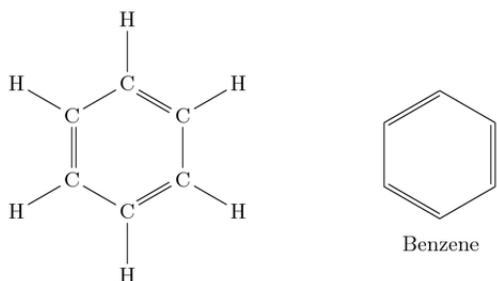


Fig. 3.4: Estructura del benceno con sus enlaces dobles y simples alternados [12].

- **Estereoisomería:** Se denomina *isómeros* a aquellas moléculas que contienen la misma fórmula química (en el sentido de proporción de átomos) pero presentan estructuras y propiedades diferentes, por ejemplo el alcohol etílico y el éter dimetílico cuya formula molecular es C_2H_6O son isómeros. Dicho esto, un *estereoisómero* es un isómero que a su vez contiene los mismos enlaces y la misma estructura de átomos enlazados pero se diferencia en la rotación u orientación tridimensional de los átomos. Los tipos de isomería modelados consisten en aquellos llamados *cis – trans* y *E/Z* que se utilizan para indicar las posiciones relativas de los sustituyentes en la molécula.

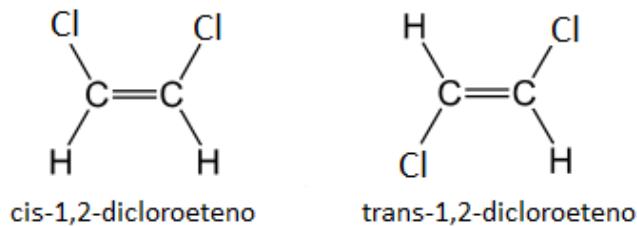


Fig. 3.5: Isómeros *cis* y *trans* del 1,2-dicloroeteno, la estructura y la fórmula son iguales pero difieren en la orientación de los átomos de cloro (Cl) [13].

Posibles valores: STEREONONE, STEREOZ, STEREOE, STEREOCIS, STEREO-TRANS

Atributos de los Átomos

Estos features contienen la información relevante del átomo en la molécula, algunos de ellos como el símbolo o la masa son propios del átomo y otros como la aromaticidad o la carga formal están relacionados a las propiedades del átomo en relación a la molécula de la que forma parte, lo mismo sucede para los atributos de los enlaces. Los posibles valores

están restringidos a la presencia de estos en el dataset utilizado, lo que implica que son un subconjunto de aquellos que pueden presentarse en otras moléculas y átomos del universo.

- **Símbolo:** Representa el átomo en cuestión con su abreviatura estándar, puede ser cualquier elemento de la tabla periódica (por ejemplo *O*, *C*, *S*, *Na*, etc).
Posibles valores: Ac, Ag, Al, Ar, As, Au, B, Ba, Be, Bi, Br, C, Ca, Cd, Cl, Co, Cr, Cs, Cu, F, Fe, Ga, Gd, Ge, H, He, Hg, I, K, Li, Mg, Mn, N, Na, Nb, Ni, O, P, Pb, Pd, Po, Pt, Rb, Rf, Rh, Ru, S, Sb, Se, Si, Sn, Sr, Tb, Tc, Ti, Tl, U, V, W, Y, Zn, Zr.
- **Grado:** El grado de un átomo se define como el número de átomos vecinos directamente enlazados, es equivalente a la definición de grado de un nodo en un grafo no dirigido.
Posibles valores: 0, 1, 2, 3, 4, 5.
- **Hibridación:** La hibridación describe los enlaces atómicos desde la perspectiva del átomo y proporciona información sobre la superposición de los orbitales atómicos en las uniones entre dos o más átomos de modo que se minimice el nivel de energía. Esta directamente relacionado con la geometría del átomo y su disposición en el espacio.
Posibles valores: *s*, *sp*, *sp*², *sps*, *sp*³*d*
- **Carga Formal:** La carga formal de un átomo es la carga relativa que presenta en la molécula asumiendo que los electrones que participan en los enlaces se comparten equitativamente. Es una medida hipotética y no representa la carga real del átomo como entidad aislada. Se suele obtener mediante la siguiente formula:

$$Q_f = V - L - \frac{1}{2}B$$

Donde *V* es el número de electrones de valencia del átomo, *L* es el número de electrones de valencia del átomo que no participan en ningún enlace y *B* es el número total de electrones compartidos en enlaces con otros átomos de la molécula.

Posibles valores: -2, -1, 0, 1, 2, 3, 4

- **Número de electrones desapareados:** Se llama electrón desapareado a aquel que no tiene su espín compensado por otro electrón de espín opuesto en el mismo átomo. Por lo tanto, se encuentra solo en un orbital, el cual se dice que está semiocupado.
Posibles valores: 0, 1, 2, 3, 4
- **Aromaticidad:** La aromaticidad es una característica de los hidrocarburos cíclicos conjugados en la que los electrones de los enlaces dobles están libres y pueden moverse de un enlace a otro, ya sea doble o simple. Esto le da a la molécula una estabilidad mayor y la hace más difícil de romper. Este atributo nos indica si el átomo pertenece a un anillo aromático.
Posibles valores: True, False
- **Pertenencia a anillo:** Indica si el átomo se encuentra en un anillo o no.
Posibles Valores: True, False
- **Masa atómica:** Es la masa del átomo expresada en unidades de masa atómica. El atributo fue normalizado para los datasets de los distintos experimentos realizados.

■ **Quiralidad:**

Algunas moléculas se presentan en dos formas distintas tal que una es la imagen espejada de la otra. A este fenómeno se lo llama *quiralidad*. Una molécula quiral cuya orientación se da hacia la derecha se la denota con la letra *R* mientras que la que se orienta hacia la izquierda se la denota con la letra *S*. Una característica de estas moléculas es que a pesar de tener la misma fórmula y estructura sus imágenes no son superponibles.

Posibles valores: R, S.

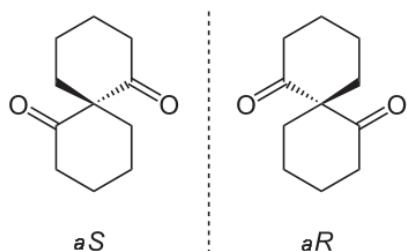


Fig. 3.6: Variantes quirales de un compuesto, *S* a izquierda y *R* a derecha [14].

3.4. Pasaje a grafo

Para conformar el grafo utilizamos la función de Chem *GetBonds()* que retorna la lista de adyacencia indicando los pares de átomos conectados. Luego los atributos de los ejes y de los nodos se codificaron en matrices separadas de dimensiones $|E| \times E_f$ y $|V| \times V_f$ donde E_f y V_f equivalen a la cantidad de atributos de los ejes y los nodos respectivamente, como vimos en la introducción de grafos. En estas matrices cada fila se corresponde con un eje o un nodo y en las columnas se encuentran los valores de cada feature asociado a ese eje o nodo. Para generar estas matrices utilizamos funciones auxiliares a modo de obtener los one-hot encodings o normalizar los atributos numéricos.

Una vez obtenidos los features de la molécula, se instancia un objeto *Data*, perteneciente al módulo *torch_geometric*, que se utiliza específicamente para contener al grafo y a los atributos de los ejes y nodos. Este objeto es el input de los los modelos que fueron entrenados. Veamos un ejemplo reducido a modo de ilustración, si queremos obtener los features para la molécula del H_2O ejecutamos los siguientes pasos:

Algoritmo 3.2: Pasaje de InChi a grafo

```
from Chem.rdinchi import InchiToMol

mol = InchiToMol('InChI=1S/H2O/h1H2') # pasamos de InChI a molcula
ady_list = mol.GetBonds() # obtenemos lista de adyacencia
edge_feats = edge_attributes(mol) # matriz de features de los ejes
node_feats = node_attributes(mol) # matriz de features de los atomos

# por ultimo instanciamos un objeto de Pythorch Geometric con los datos
data = torch_geometric.data.Data(x=node_feats,
                                  edge_index=ady_list,
                                  edge_attr=edge_feats,
                                  y=label) # y corresponde a la clase,
                                             negativa o positiva
```

De esta forma construimos los objetos que utilizan los modelos para entrenar y predecir las propiedades de la molécula. El atributo *y* del objeto *Data* corresponde a la clase, en un clasificador binario puede ser 1 o 0 mientras que en los modelos multiclas se compone de una lista de ceros y unos cuya *i*-ésima posición indica si la molécula se asocia con la *i*-ésima clase. Las funciones auxiliares *edge_attributes()* y *node_attributes()* procesan la molécula y retornan las matrices con los features listas para ser utilizadas por el modelo.

4. REDES DE GRAFOS CONVOLUCIONALES

En el transcurso de las décadas pasadas, las redes neuronales artificiales y, específicamente las redes convolucionales (CNN) y redes recurrentes (RNN), tuvieron un profundo impacto en varias aplicaciones de la inteligencia artificial, por ejemplo en áreas relacionadas al reconocimiento de imágenes y al procesamiento del lenguaje, en gran medida estos progresos fueron alcanzados en dominios de representación basados en secuencias e imágenes. Una característica positiva de estos dominios es que pueden ser codificados en estructuras de tipo grilla, ya sea como listas o arreglos bidimensionales en el caso de imágenes, esto permite representar la información en espacios Euclídeos beneficiándose de las propiedades del álgebra lineal para operar con los datos. Sin embargo, existen múltiples campos donde la información no puede ser trivialmente embebida en un espacio Euclídeo pero sí puede ser representada naturalmente con **grafos**, ejemplo de esto son las redes sociales, las redes de telecomunicación, y por sobre todo, las moléculas. Esto explica el interés surgido en la comunidad científica en los últimos años por generar nuevas tecnologías capaces de operar en este tipo de dominios de estructura irregular con el fin de mejorar el estado del arte en el que se encuentra la inteligencia artificial. En este contexto son múltiples las arquitecturas de redes neuronales de grafos (GNN) que han sido propuestas e implementadas.

Las GNN son redes neuronales que funcionan generando representaciones internas de los nodos al considerar la información de la estructura completa del grafo, en contraposición a las arquitecturas convolucionales clásicas donde la arquitectura de la red se encuentra influenciada por la conocida a priori e invariante estructura de los datos (matrices bidimensionales para las imágenes por ejemplo). Dado que por naturaleza los grafos poseen una estructura irregular, las GNN propagan la información de los nodos a través de la red de acuerdo a la topología particular de cada grafo. Entre las GNN, las GNN convolucionales (GCN) apuntan a simular el método utilizado por las CNN para extraer features a través de matrices de pesos compartidas. En las imágenes la operación de convolución funciona aplicando una suma ponderada de los atributos de los nodos vecinos de cada píxel con una matriz llamada *kernel*, lo cual es posible gracias a que las posiciones relativas en que se encuentran los píxeles es conocida previamente, en cambio, en las estructuras representadas con grafos, este tipo de convoluciones no resulta trivial dado que primero existe un número variable y no acotado de vecinos posibles, por lo que no es posible utilizar filtros de longitud fija en las convoluciones, en segundo lugar no existe orden relacionado a la distribución de los nodos vecinos. Es por esto que es necesario redefinir el operador de convolución al diseñar este tipo de redes.

Las convoluciones en grafos guardan cierta semejanza con las convoluciones en imágenes en el sentido que parte del proceso consiste en la multiplicación de una matriz de pesos con los atributos de los nodos, seguido por algún tipo de agregación como puede ser una suma o promedio ponderado. Al mismo tiempo, así como en las convoluciones en imágenes los atributos de los nodos vecinos son “compartidos” en el cómputo de la convolución, las GCN utilizan un mecanismo llamado **pasaje de mensajes** que permite a los nodos conocer el vector de features de sus nodos vecinos de forma que puedan compartir la información. Este tipo de redes de grafos convolucionales fue introducido por Kipf et al en 2016 y siguen

siendo ampliamente estudiadas en la actualidad, para comprender la motivación y los mecanismos que hacen posible su funcionamiento, resulta primordial comenzar hablando de las convoluciones en su formulación clásica.

4.1. Redes convolucionales

Las redes convolucionales concebidas en 1989 por LeCun son un tipo de arquitectura neuronal especialmente diseñadas para procesar información estructurada en forma de grilla, ejemplo de esto pueden ser series de datos temporales como el precio diario de una acción, representado como un arreglo unidimensional, o por ejemplo, imágenes representadas a través de matrices bidimensionales de píxeles. El nombre “red convolucional” proviene de la utilización de un operador matemático llamado **convolución**, y este tipo de redes se caracteriza por utilizar esta operación en al menos una de sus capas en contraposición de la multiplicación clásica de matrices. Usualmente la operación usada en estas redes no se corresponde exactamente con la definición usada en las matemáticas puras pero se encuentra fuertemente relacionada.

En el dominio discreto, la convolución de dos funciones f y g de una variable se denota típicamente con un asterisco como:

$$s(t) = (f * g)(t) = \sum_{a=-\infty}^{\infty} f(a)g(t-a)$$

En el campo de las redes neuronales se considera a la función f como el *input* y a la función g como el *kernel*, usualmente el input suele ser un arreglo multidimensional de datos y el kernel un arreglo multidimensional de parámetros que son adaptados por el algoritmo de aprendizaje. Esta definición de convolución puede extenderse también para funciones de dos dimensiones, si consideramos un dominio finito acotado en la primera dimensión por m y en la segunda por n podemos expresar la fórmula como:

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n)$$

Dado que la convolución tiene la propiedad de ser conmutativa podemos reescribir la fórmula como:

$$S(i, j) = (K * I)(i, j) = \sum_m \sum_n I(i - m, j - n)K(m, n)$$

Estas operaciones son ampliamente utilizadas en el procesamiento digital de imágenes, donde I suele ser una imagen representada como una matriz de píxeles en dos dimensiones y K suele ser una matriz más pequeña llamada *filtro*, de 3x3 por ejemplo, que al ser convolucionada sobre la imagen es capaz de desenfocarla, detectar bordes o realzarla entre tantas otras posibilidades.

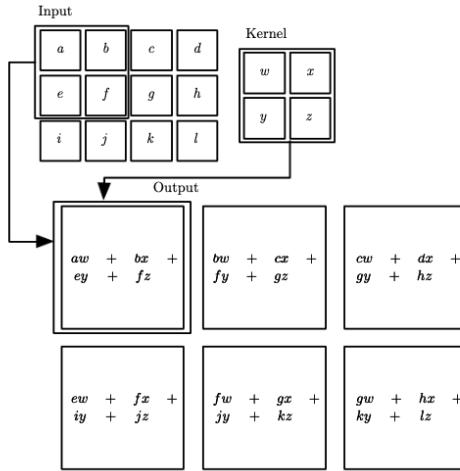


Fig. 4.1: Ejemplo de un kernel (matriz superior derecha) aplicado a una matriz (superior izquierda) [23].

Si aplicamos el siguiente filtro a una imagen este generara una simple detección de bordes:

$$K = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$



Fig. 4.2: Imagen original y después de aplicar el filtro [15].

Las convoluciones gozan de ciertas propiedades importantes que benefician a los algoritmos de aprendizaje:

- **Pesos esparsos:** dado que la operación de convolución utiliza kernels pequeños en comparación con el input en vez de utilizar la multiplicación clásica de matrices donde todas las unidades de salida interactúan con todas las de entrada, la red es capaz de detectar features y propiedades utilizando menos espacio para almacenar los filtros.

- **Intercambio de parámetros:** hace referencia a que todas las componentes del kernel son usadas en todas las posiciones del input, a diferencia de las capas densas donde cada peso se utiliza una vez para generar la capa de salida , por lo que en vez de aprender un conjunto de parámetros para cada sección del input se aprende uno solo para todo el conjunto.
- **Equivarianza traslacional:** es una propiedad de las convoluciones que a grandes rasgos permite mantener la representación a pesar de traslaciones en el input. Si por ejemplo estamos construyendo una red para detectar autos en imágenes, los mismos features deberían aparecer representados en una imagen que contiene un auto y en otra donde ese mismo auto se encuentra unos centímetros o píxeles desplazado hacia la derecha.

Típicamente las capas convolucionales consisten en tres etapas de cómputo secuencial, la primera consiste en aplicar una serie de convoluciones en paralelo con distintos filtros que generan el conjunto de activaciones lineales. En segundo lugar estas activaciones pasan por funciones de activación no lineales como *ReLU*, finalmente se aplica una función de **pooling** que consiste en reemplazar el output de la red en ciertos puntos con alguna operación de agregación aplicada al entorno de ese punto. Por ejemplo **max pooling** retorna el valor máximo en un entorno rectangular de puntos.

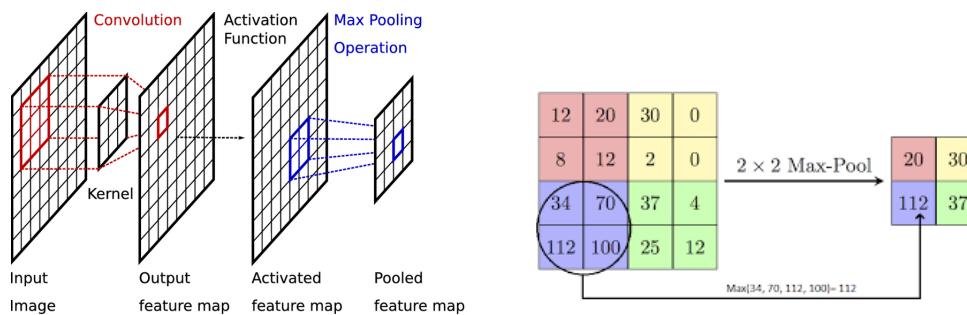


Fig. 4.3: A izquierda las tres etapas de una capa convolucional (convolución, función de activación y pooling) y a derecha la operación de max pooling [16][17].

Las capas de pooling suelen ayudar a que la representación generada sea invarianta a pequeños desplazamientos en el input, ayudando a detectar cuando un feature esta presente sin darle tanta importancia a la ubicación específica en la que se encuentra.

Una aspecto a tener en cuenta es que estas redes están diseñadas para trabajar con datos estructurados como grilla, ya sean arreglos de una, dos o más dimensiones, las grillas se caracterizan por tener una estructura fija definida, en el caso de una lista de valores todos los elementos se caracterizan por tener dos elementos vecinos, el anterior y el siguiente, salvo por los elementos de los extremos, en el caso de una imagen, todos los píxeles de la matriz tienen 8 píxeles vecinos, salvo por aquellos que se encuentran en los bordes. Estas propiedades estructurales de las grillas impiden que las convoluciones puedan ser utilizadas en grafos de la misma forma, ya que una de las propiedades de los grafos es su geometría variable dado que hay grafos que no son conexos, otros que son completos y otros cuyos nodos tienen grados distintos y por lo tanto un número de nodos vecinos variable. Es

por esto que la convolución necesita ser redefinida de forma tal que pueda operar sobre datos cuya estructura sea irregular, como veremos a continuación las **redes de grafos convolucionales** dan una solución a este problema.

4.2. Convoluciones en grafos

Consideremos un grafo $G = (V, E)$ con $|V| = N$ donde cada nodo tiene asociado un vector con D features y sea $H^{(0)} \in \mathcal{R}^{N \times D}$ su matriz asociada cuya i -ésima fila contiene el vector de features correspondientes al nodo i . Definimos la convolución a través del siguiente operador:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}) \quad (4.1)$$

Donde $H^{(l)}$ es la matriz de activaciones de la capa l , $\tilde{A} = A + I_N$ es la matriz de adyacencia de G con la identidad añadida, \tilde{D} es una matriz diagonal donde \tilde{D}_{ii} contiene la cantidad de vecinos del nodo i y $W^{(l)}$ es la matriz de pesos entrenable de la capa l . Finalmente σ denota una función de activación como puede ser $ReLU(\cdot)$.

Para entender el funcionamiento de la ecuación anterior podemos desglosarla y comprenderla por partes. En primer lugar, el hecho de multiplicar el vector de features $H^{(l)}$ a izquierda por la matriz de adyacencia \tilde{A} permite compartir los vectores de features de los nodos vecinos entre si al generar una nueva matriz que contiene en cada fila la suma de los features de un nodo con los de sus vecinos, veamos un ejemplo con ambas matrices:

$$\tilde{A} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \quad H = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad \tilde{A} \times H = \begin{bmatrix} 5 & 7 & 9 \\ 12 & 15 & 18 \\ 11 & 13 & 15 \end{bmatrix}$$

En este ejemplo \tilde{A} representa un grafo cuyos ejes están dados por $E = \{(1,1), (1,2), (2,2), (2,3), (3,3)\}$ y H su matriz de features. Al multiplicar ambas matrices, la matriz resultante contiene en su i -ésima fila la i -ésima fila de H sumada con las filas de H asociadas a los vecinos del nodo i , dicha información viene codificada en la matriz de adyacencia. En la definición que utilizamos en vez de usar la matriz de adyacencia se utiliza una versión normalizada del Laplaciano del grafo dada por:

$$\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$$

De esta forma el resultado no consiste solamente en la suma de los vectores de los nodos vecinos si no que es un promedio ponderado por el grado de los nodos, evitando de esta forma inestabilidades numéricas y desvanecimientos o explosiones del gradiente[38]. Si por lo tanto consideramos la siguiente parte del operador:

$$\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)}$$

Podemos interpretar que en esta operación está ocurriendo un **pasaje de mensajes** en donde los nodos del grafo comparten con sus vecinos la información que contienen ponderando los features que reciben con los parámetros de normalización mencionados. La siguiente imagen ilustra este concepto:

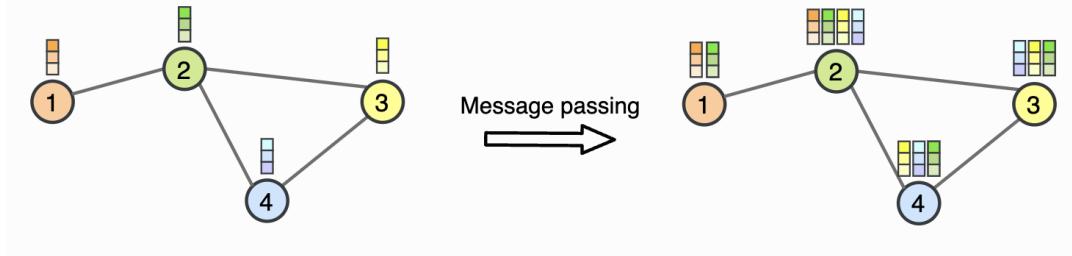


Fig. 4.4: Cada nodo recibe la información de sus nodos vecinos [6].

Esto nos retrotrae a las capas convolucionales explicadas en el apartado anterior, donde el output para una posición dada se generaba con la información de los píxeles circundantes. Dicha analogía puede verse simplificada en la siguiente ilustración.

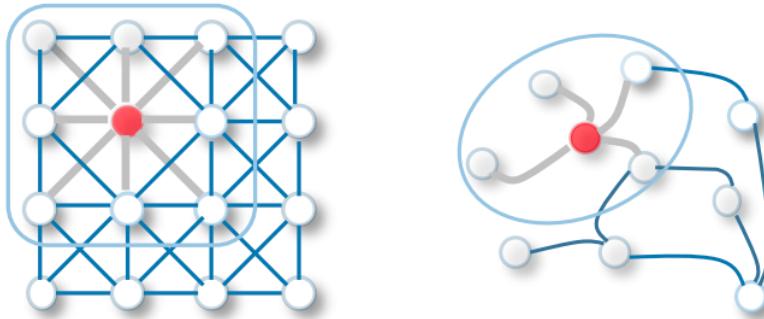


Fig. 4.5: A izquierda se puede apreciar que la convolución clásica sobre una estructura de tipo grilla sobre el nodo rojo alcanza exactamente a sus nodos circundantes mientras que en la convolución de la derecha, aplicada sobre el nodo rojo en una estructura irregular, los nodos circundantes consisten en los vecinos directos del nodo [18].

Casi por último tenemos a derecha una multiplicación por una matriz $W^{(l)} \in \mathcal{R}^{D \times O}$, siendo D el número de features que contiene cada nodo en la capa l , dado por la dimensión de las columnas de la matriz $H^{(l)}$ y O la dimensión de los features resultantes, muchas veces llamado “canales de salida”. $W^{(l)}$ será la matriz de pesos que aprenderá el algoritmo en las sucesivas iteraciones del entrenamiento y permite variar arbitrariamente la cantidad de features de salida que deseamos para cada nodo. Finalmente, al igual que en las capas convolucionales clásicas se aplica una función de activación σ para introducir la no linealidad. De esta forma, luego de aplicar el operador convolucional, se obtiene una matriz $H^{(l+1)} \in \mathcal{R}^{N \times O}$.

Al igual que con las capas convolucionales, las convoluciones en grafos pueden aplicarse sucesivamente para lograr arquitecturas más profundas ya que con cada capa que se agrega

a la red se propaga en un salto más la información de los nodos vecinos, de este modo, con la aplicación de una primer capa convolucional los nodos solo serán capaces de ver la información de sus nodos vecinos, pero si se agrega una segunda capa, la información fluirá entre todos los nodos que se encuentren a dos saltos de distancia, permitiendo que los features fluyan a través de toda la red que forma el grafo de a un salto por capa, lo que puede resultar muy útil en redes de grafos con caminos máximos de gran longitud. Un punto a destacar es que la featurización obtenida luego de las sucesivas aplicaciones de capas convolucionales genera una nueva matriz de features con tantas filas como nodos y tantas columnas como cantidad de features se hayan especificado en el canal de salida de la última convolución. Lo que se suele hacer es aplanar la matriz en un vector al aplicar una operación de **pooling** que sume, tome el máximo o el promedio de cada columna en la matriz, obteniendo de esta forma un único vector de salida que puede usarse como input para las siguientes capas densas de la red.

El operador convolucional que presentamos es una de las bases sobre la cual se han ido construyendo y explorando muchas variantes posibles que dieron nacimiento a convoluciones más complejas, entre ellas algunas incorporan mecanismos de **atención** y permiten incorporar **features en los ejes** del grafo por ejemplo, algunas de esas arquitecturas fueron exploradas y utilizadas en esta tesis y las veremos a continuación.

Graph Attention Networks

Las redes de grafos con atención (GAT) son una variante de las redes de grafos convolucionales, fueron introducidas por Velickovi et al [31] en 2018 y se caracterizan por incorporar un **mecanismo de atención** entre las conexiones de los nodos de forma que la red pueda aprender cuan relevante es la información de un nodo j para un nodo vecino i , introduciendo la noción de que **los nodos no deben tener todos la misma importancia**. Similar a las GCN, las GAT utilizan un mecanismo de pasaje de mensajes entre nodos compuesto por una matriz de pesos utilizada para generar el vector de features resultantes que, a diferencia de las GCN, luego es reescalado por un factor α_{ij} que denota la importancia de esa conexión entre el nodo i y el nodo j . Como veremos, ese factor es un parámetro entrenable que se ajusta durante el entrenamiento.

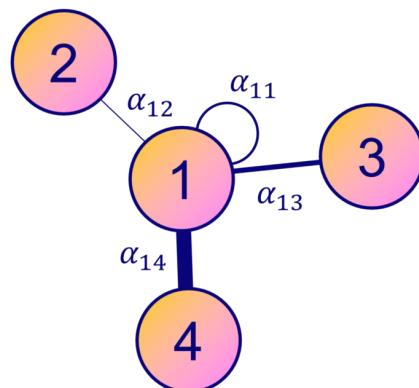


Fig. 4.6: Las conexiones entre los nodos tienen distintos grados de prioridad, ilustrado por el grosor de los ejes [19].

Formalmente, las GAT son capas que reciben como input el conjunto de features de los nodos, $\mathbf{h} = \{\overrightarrow{h_1}, \overrightarrow{h_2}, \dots, \overrightarrow{h_N}\}$, $\overrightarrow{h_i} \in R^F$ siendo N la cantidad de nodos, y luego retornan un nuevo conjunto de features para todos los nodos, $\mathbf{h}' = \{\overrightarrow{h'_1}, \overrightarrow{h'_2}, \dots, \overrightarrow{h'_N}\}$, $\overrightarrow{h'_i} \in R^{F'}$, potencialmente de distinta cardinalidad, al igual que en las GCN.

El operador de las capas con atención puede interpretarse como una secuencia de tres pasos sucesivos:

1. **Transformación lineal**
2. **Función de activación no lineal**
3. **Normalización softmax**

En el primer paso se aplica una transformación lineal mediante una matriz $\mathbf{W} \in \mathcal{R}^{F' \times F}$ a todos los nodos, luego se utiliza un mecanismo de atención $a : \mathcal{R}^{F'} \times \mathcal{R}^{F'} \rightarrow \mathcal{R}$ que computa los coeficientes de atención, indicando la importancia de los features del nodo j al nodo i , notamos esa transformación así:

$$e_{ij} = a(\mathbf{W} \overrightarrow{h_i}, \mathbf{W} \overrightarrow{h_j}) \quad (4.2)$$

Para no perder la información estructural del grafo, los coeficientes e_{ij} se computan para los nodos $j \in N_i$, donde N_i es el conjunto de vecinos del nodo i . El mecanismo de atención a consiste en una red feedforward de una sola capa parametrizada por un vector $\vec{\mathbf{a}} \in R^{2F'}$ con una activación no lineal dada por la función LeakyReLU (paso 2). De esta forma el coeficiente puede reescribirse como:

$$e_{ij} = \text{LeakyReLU}(\vec{\mathbf{a}}^T [\mathbf{W} \overrightarrow{h_i} || \mathbf{W} \overrightarrow{h_j}]) \quad (4.3)$$

Donde $||$ denota la concatenación de los vectores. Por último, el tercer paso consiste en aplicar la normalización softmax para que los coeficientes sean comparables en la misma escala a través de los nodos del grafo, de modo que el coeficiente final queda definido como:

$$\alpha_{ij} = \text{softmax}(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in N_i} \exp(e_{ik})} \quad (4.4)$$

Expandiendo la fórmula podemos expresar el coeficiente del siguiente modo:

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(\vec{\mathbf{a}}^T [\mathbf{W} \overrightarrow{h_i} || \mathbf{W} \overrightarrow{h_j}]))}{\sum_{k \in N_i} \exp(\text{LeakyReLU}(\vec{\mathbf{a}}^T [\mathbf{W} \overrightarrow{h_i} || \mathbf{W} \overrightarrow{h_k}]))} \quad (4.5)$$

En la siguiente imagen vemos una ilustración que describe los pasos involucrados en el cómputo de los coeficientes, de izquierda a derecha se encuentran la transformación lineal aplicada a los features de los ejes, la activación no lineal y por último la normalización.

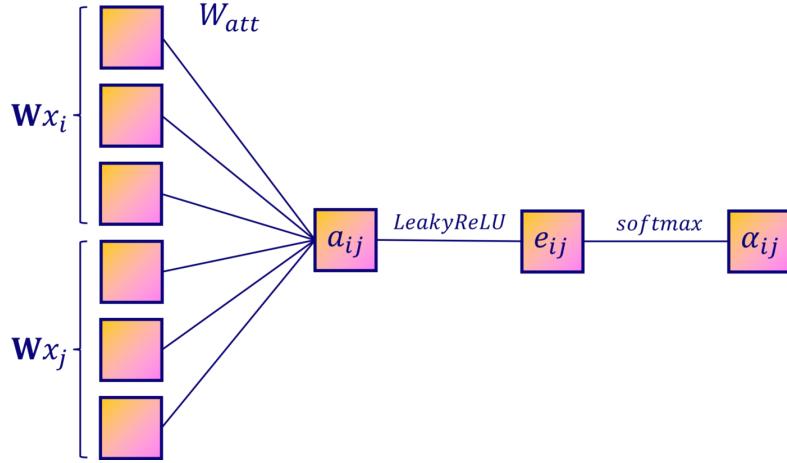


Fig. 4.7: Etapas en el cálculo de coeficientes [19].

Finalmente los coeficientes de atención son utilizados para ponderar la importancia de los features entre nodos vecinos, por lo tanto podemos definir el vector de features de salida para el nodo i como:

$$\vec{h}'_i = \sigma \left(\sum_{j \in N_i} \alpha_{ij} \mathbf{W} \vec{h}_j \right) \quad (4.6)$$

Para estabilizar el proceso da aprendizaje también es posible utilizar mecanismos *multi – head* combinando la ecuación anterior a través de K mecanismos independientes cuyos resultados finales pueden ser o bien concatenados obteniendo así la expresión:

$$\vec{h}'_i = \left\| \sum_{k=1}^K \sigma \left(\sum_{j \in N_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j \right) \right\| \quad (4.7)$$

o bien promediados:

$$\vec{h}'_i = \sigma \left(\frac{1}{K} \sum_{k=1}^K \sum_{j \in N_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j \right) \quad (4.8)$$

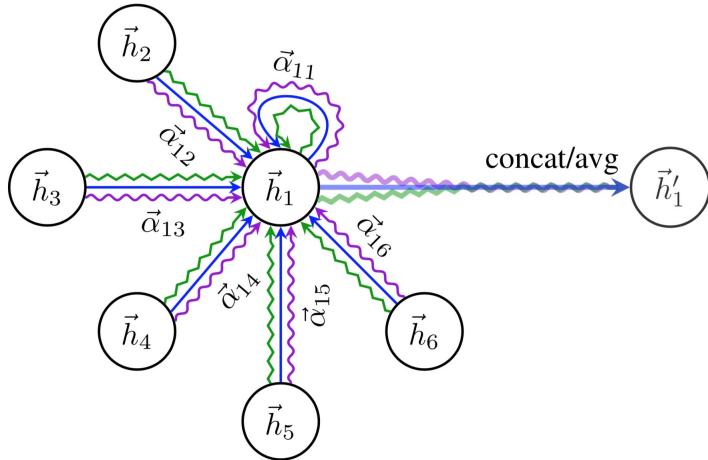


Fig. 4.8: Ejemplo del cálculo de coeficientes con multi head attention ($K=3$). Los coeficientes de atención se calculan para el nodo 1 con la información de sus nodos vecinos, las distintas flechas denotan los tres coeficientes independientes [31].

Codificando atributos de los ejes

En las convoluciones recién presentadas no se toman en consideración los posibles atributos de los ejes, sin embargo en las GAT es posible añadirlos en el cómputo de los coeficientes de atención mediante la multiplicación de otra matriz de pesos \mathbf{W}_e y la posterior concatenación con los vectores de features de los nodos como podemos apreciar en la siguiente ecuación:

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(\vec{\mathbf{a}}^T [\vec{\mathbf{W}} \vec{h}_i || \vec{\mathbf{W}} \vec{h}_j || \vec{\mathbf{W}}_e \vec{e}_{ij}]))}{\sum_{k \in N_i} \exp(\text{LeakyReLU}(\vec{\mathbf{a}}^T [\vec{\mathbf{W}} \vec{h}_i || \vec{\mathbf{W}} \vec{h}_k || \vec{\mathbf{W}}_e \vec{e}_{ik}]))} \quad (4.9)$$

Las GCN y las GAT pueden considerarse pilares fundamentales dentro de las arquitecturas convolucionales existentes hoy en día, ya que sentaron bases sólidas para el desarrollo y el estudio de este tipo de redes que recién se encuentran en una fase temprana y de exploración. En este trabajo implementamos y experimentamos con ambas arquitecturas y una de sus derivadas, las redes *AttentiveFP*, cuyo mecanismo se basa en las redes GAT pero con ciertas modificaciones del operador convolucional. En la etapa de experimentación detallaremos los parámetros y la estructura de las redes que utilizamos en los modelos.

Attentive FP

Las redes Attentive FP [40] extienden un paso más el mecanismo de atención de las graph attention networks y buscan capturar patrones intramoleculares más allá del entorno

local de cada nodo. Sin embargo a diferencia de las capas GCN o GAT, las Attentive FP pueden considerarse como una arquitectura de red en si misma más que un simple mecanismo de convolución para el pasaje de mensajes. Para esto generan los embeddings de los átomos utilizando un mecanismo de atención que combina el uso de capas GAT con gated recurrent units (GRU) y luego computan un vector de features asociado a la molécula combinando los vectores de atributos de cada nodo mediante la suma (global add pool). Finalmente se aplica de nuevo el mismo mecanismo de atención utilizado para los embeddings de los átomos sobre el vector de features de la molécula el cual alimenta una capa fully connected que computa el output final de la red. El cómputo en las redes attentive puede resumirse en la siguiente serie de pasos:

1. Se computan los embeddings de los nodos mediante una capa fully connected.
2. Aplicación del mecanismo de atención(GAT) sobre los features de los nodos.
3. Aplicación de una gated recurrent unit (GRU) entre los embeddings del nodo generados en el paso 1) y el output de la GAT del paso 2).
4. Embedding de la molécula mediante un add pool con los vectores de los nodos.
5. Aplicación del mecanismo de atención GAT al embedding de la molécula.
6. Aplicación de la GRU entre el embedding de la molécula del paso 4) y el output de la GAT del paso 5).
7. Cómputo del output final mediante una capa fully connected.

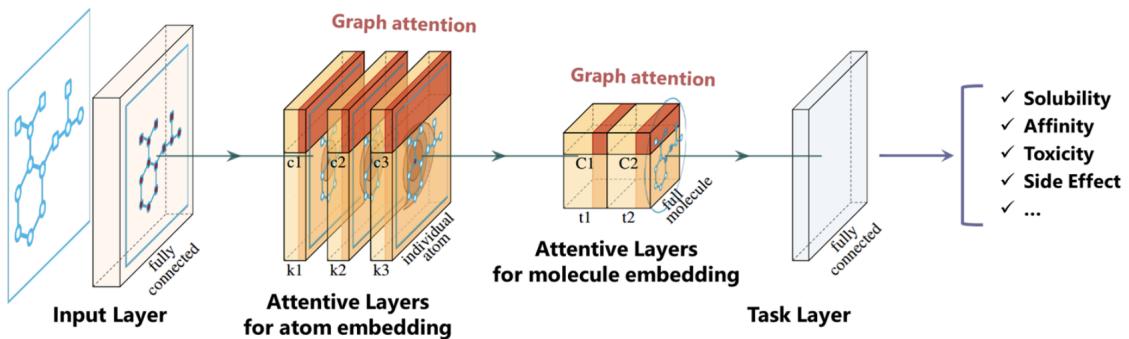


Fig. 4.9: Arquitectura de la red attentive [40].

Los pasos 2 y 3 que consisten en aplicar el mecanismo de atención y luego pasar por la gated recurrent unit pueden considerarse como un único módulo o capa que puede ser aplicada reiteradamente para extraer features de los átomos, el número de iteraciones que se aplican estos pasos es un parámetro del modelo. Puede observarse que los pasos 5 y 6 son iguales, esto es porque el mismo mecanismo se aplica luego al embedding de la molécula con el fin de extraer patrones no locales que pueden contener información relacionada a pares de nodos que no son necesariamente vecinos. La cantidad de veces que se aplica el mecanismo al embedding de la molécula también es un parámetro del modelo. En la figura anterior

se aplican tres iteraciones a los embeddings de los nodos y dos iteraciones al de la molécula.

El vector de features de un nodo v luego de k iteraciones del paso 3) queda definido como:

$$h_v^k = GRU^{k-1}(C_v^{k-1}, h_v^{k-1}) \quad (4.10)$$

Donde C_v^{k-1} representa el vector de salida generado por el mecanismo de atención y h_v^{k-1} es el vector de features de la iteración anterior.

Tipos de tareas clasificación

Las tareas sobre datos estructurados como grafos pueden ser agrupadas en tres categorías: **nivel de nodos**, **nivel de ejes**, **nivel de grafos**. Cada categoría describe sobre qué nivel se quiere hacer la clasificación o regresión. Veamos un ejemplo para cada caso.

El **nivel de nodos** tiene por objetivo clasificar nodos pertenecientes a un mismo grafo. Por ejemplo el dataset *Cora*[20] consiste en un grafo que representa una red de citaciones entre papers de distintas clases donde los papers vienen a constituir los nodos del grafo y, cuando un paper cita a otro, existe un eje entre los nodos del grafo que representan ambos papers respectivamente. Los features de los nodos consisten en un bag of words de 1.433 palabras, o sea, un vector de 1.433 elementos donde la i -ésima posición indica con un 1 que la palabra se encuentra presente en el paper y un 0 indica su ausencia. A su vez cada paper puede ser clasificado en una de entre siete categorías. El objetivo es entrenar un clasificador que pueda inferir la categoría de algunos papers dentro del dataset para los cuales el modelo no conoce a cual de las siete pertenece. Para esto puede entrenarse una GNN de forma semi supervisada donde la red solo conoce la categoría de un subconjunto de los papers y el resto será inferida por el modelo utilizando la estructura de la red dada por el grafo y el vector de features de los nodos dado por la bag of words.

En algunos contextos es requisito saber si en un grafo dado, existe o debería existir un eje entre dos nodos. Este tipo de tareas se asocia al **nivel de ejes**. Imaginemos una red social, Facebook por ejemplo, donde los usuarios pueden pensarse como los nodos de la red y la relación “ser amigo de” entre dos usuarios implica que existe un eje entre los nodos respectivos del grafo. Una funcionalidad muy común en este tipo de redes es la sugerencia de “amigos”, donde hay un algoritmo que nos está sugiriendo constantemente nuevos amigos para hacer dentro de la red, o a nivel del grafo, nos está sugiriendo que debería o podría haber un eje entre dos nodos.

Finalmente el **nivel de grafos** consiste en clasificar la estructura completa de un grafo en vez de clasificar nodos o ejes dentro de un solo grafo. Para esto debemos contar con un dataset que contenga múltiples grafos a clasificar basados en sus propiedades estructurales. La tarea más común suele ser la clasificación de propiedades moleculares, dicho sea de paso, es el objetivo que perseguimos en este trabajo. En este tipo de problemas los nodos representan los átomos de las moléculas y los enlaces son representados por los ejes del grafo. De esta forma los clasificadores predicen la pertenencia de un grafo/molécula a alguna categoría, en nuestro caso relacionada a sabores.

5. EXPERIMENTACIÓN

5.1. Software y Metodología

Para construir los modelos y generar instancias parametrizables de experimentos desarrollamos un software basado en los módulos de machine learning **PyTorch** y **PyTorch Geometric** de Python, PyTorch Geometric contiene las implementaciones de las capas convolucionales mencionadas en el apartado anterior. Sobre estos módulos implementamos un programa que nos permitió procesar los datasets para pasar de InChis a grafos añadiendo los atributos de los enlaces y los átomos y subsecuentemente incorporarlos en el tipo de datos utilizado por PyTorch Geometric, representando los grafos como listas de adyacencia y los atributos con matrices en cuyas filas se codificaron los atributos de los nodos y ejes, utilizando matrices distintas para ambos. Por otro lado el software permite construir y ejecutar instancias aleatorias de experimentos parametrizando tanto los hiperparámetros de la red como su arquitectura. Esto último permite definir en un archivo de configuración distintas estructuras de red al variar la cantidad y la configuración de las capas densas y convolucionales con el fin de encontrar la arquitectura que mejor se adapta a los datos provistos sin la necesidad de escribir manualmente el código de cada arquitectura con la que se quiera experimentar. En particular optamos por experimentar utilizando entre una y tres convoluciones variando el número de canales de salida entre 64 y 128, la salida del módulo convolucional alimenta una red feedforward cuya profundidad oscila entre una y tres capas densas. En cuanto a las capas convolucionales, utilizamos las dos variantes descritas en la literatura y también redes Attentive FP:

- **graph convolutional networks** (GCN): Son capas en donde la convolución está definida mediante la ecuación 4.1 de la sección 4.2. Este tipo de redes no permite codificar los atributos de los ejes, los detalles técnicos pueden encontrarse en la [documentación](#) de PyTorch Geometric.
- **graph attention networks** (GAT): Son capas en donde la convolución está definida mediante la ecuación 4.6 de la sección 4.2. Este tipo de redes sí permite codificar los atributos de los ejes, a diferencia de las GCN, los detalles técnicos pueden encontrarse en la [documentación](#) de PyTorch Geometric.
- **attentive fp** (ATTENTIVE): Al igual que las GAT, este tipo de redes sí permite codificar los atributos de los ejes, los detalles técnicos pueden encontrarse en la [documentación](#) de PyTorch Geometric.

Métricas utilizadas

Para simplificar las fórmulas vamos a denominar *TP* a los verdaderos positivos, o sea aquellos ejemplos que fueron correctamente clasificados como positivos por el modelo. A los falsos positivos los vamos a denominar *FP*, son aquellos ejemplos que el modelo determinó como positivos pero no lo eran. Los verdaderos negativos los simplificaremos con *TN*, son los ejemplos que el modelo clasificó correctamente como pertenecientes a la clase negativa y por último, los falsos negativos, *FN*, serán aquellos ejemplos que el modelo clasificó erróneamente como pertenecientes a la clase negativa cuando en realidad pertenecían

a la clase positiva.

Dependiendo el tipo de problema y dataset utilizado existe una variedad de métodos y métricas disponibles para medir la performance de los modelos, para evaluar clasificadores binarios en conjuntos balanceados la **exactitud** suele ser un buen indicador pero datasets no balanceados requieren métricas más específicas como **precisión** y **recall**, a continuación detallamos las métricas utilizadas que fueron consideradas en los experimentos realizados:

- **Exactitud:** mide el porcentaje de casos en los que el modelo acierta correctamente su predicción.

$$\text{exactitud} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precisión:** es el porcentaje de aciertos sobre los ejemplos que fueron clasificados como positivos.

$$\text{precisión} = \frac{TP}{TP + FP}$$

- **Recall:** es el porcentaje de ejemplos de la clase positiva que fue clasificado como positivo por el modelo.

$$\text{recall} = \frac{TP}{TP + FN}$$

- **F1:** se utiliza para combinar las métricas de precisión y recall en un solo valor, se calcula como la media armónica entre ambas métricas.

$$f1 = 2 * \frac{\text{precisión} * \text{recall}}{\text{precisión} + \text{recall}}$$

- **AUC/ROC:** la curva roc(receiver operating characteristics) es una curva de probabilidad que contiene la proporción entre el ratio de verdaderos positivos (recall) y el ratio de falsos positivos al variar el umbral de discriminación del modelo. Cuanto más cerca a uno se encuentra el área debajo de la curva roc(auc) mejor es el modelo separando las clases negativa y positiva.
- **Curva precisión-recall:** esta curva muestra como varía la precisión y el recall al variar los umbrales de discriminación.

Metodología

Para la realización de los experimentos generamos particiones de los datos separando en tres conjuntos de entrenamiento, validación y testeо, los porcentajes dependen de cada experimento en particular así como el balanceo de los datasets, luego definimos un primer espacio de búsqueda de los hiperparámetros del modelo y las convoluciones posibles y corrimos en secuencia instancias aleatorias del experimento (la cantidad depende de cada experimento). Finalmente procedimos a refinar los parámetros en base a los mejores

resultados obtenidos y modificamos el conjunto de búsqueda agregando valores entre los intervalos de parámetros que dieron mejores resultados y corrimos una nueva batería de experimentos refinados, e.g. si el learning rate en la primer secuencia de experimentos tenía asignado algún valor aleatorio entre 0.001, 0.01 y 0.1 y los mejores resultados obtenidos fueron con 0.001 y 0.01, procedimos a modificar el espacio de búsqueda agregando un valor intermedio en dicho intervalo de modo que en la nueva secuencia de experimentos el learning rate quede comprendido entre los valores 0.001, 0.005, 0.01. En todos los experimentos planteados utilizamos las tres arquitecturas convolucionales y las comparamos con la utilización de una red feedforward sin convoluciones para verificar empíricamente si estas aportan una mejora de rendimiento.

Al hacer referencia a los hiperparámetros usados en el entrenamiento de los clasificadores utilizamos las siguientes abreviaciones para referirnos a ellos:

- **l.r.** = learning rate, es el factor por el cual se escala el gradiente de la función de pérdida en el algoritmo de descenso por gradiente. Cuando el optimizador utilizado adapta el learning rate dinámicamente esta variable representa el valor inicial.
- **w.d.** = weight decay también conocida como normalización L^2 , es un parámetro de regularización de los pesos de la red.
- **d.o.** = dropout es un parámetro de normalización que elimina conexiones entre neuronas aleatoriamente en base a un factor especificado.
- **b.n** = batch normalization permite aumentar la velocidad de entrenamiento y la estabilidad de la red al normalizar el vector de entrada de las capas centrando su media y varianza.
- **opt** = optimizador, es el algoritmo de optimización utilizado para ajustar los pesos de la red, por ejemplo SGD, Adam o AdamW.
- **conv 1 y 2** de estar presentes contienen los tamaños de los canales de entrada y salida de las respectivas convoluciones.
- **f.c.#** indica la cantidad de capas fully connected de la red.
- **pool** determina la función de pooling utilizada para aplanar la matriz de features resultante del módulo convolucional.

5.2. Dulce vs Amargo

Preliminares

El primer experimento realizado consistió en entrenar clasificadores para distinguir entre moléculas dulces de moléculas amargas. Para esto generamos dos conjuntos de moléculas, aquellas que contienen la etiqueta *bitter* (amargo) y no contienen las etiquetas *sweet* ni *sweet-like* y por otro lado aquellas que contienen la etiqueta *sweet* y alguna otra que no fuera *bitter*. Además, fueron eliminadas aquellas moléculas cuya descripción de sabor tuviera referencias a la categoría opuesta independientemente de que esta no figurase en el conjunto de etiquetas. Por ejemplo, la molécula *dihydrocoumarin* contiene las siguientes etiquetas [“coconut”, “tobacco”, “coumarin”, “herbal”, “creamy”, “sweet”, “cinnamon”, “almond”, “tonka”], como contiene “sweet” debería pertenecer a la partición de moléculas dulces pero en la descripción de sabor de la molécula se encuentra la siguiente sentencia: “sweet then bitter@burning”. Como hay una referencia a la clase *bitter* esta molécula no se consideró para el conjunto de moléculas dulces. El mismo procedimiento fue llevado a cabo con las moléculas de la clase *bitter*. El dataset finalmente quedó compuesto por 572 moléculas dulces y 661 moléculas amargas.

A la etiqueta *bitter* le asignamos la clase 0 y a *sweet* la clase 1 y generamos los datasets de entrenamiento, validación y testeo tomando 70% para el primero y 15% para ambos restantes. Luego corrimos 100 experimentos en el primer espacio de búsqueda y finalmente 50 experimentos con los parámetros refinados. Finalmente procedimos a quedarnos con los modelos más precisos para cada arquitectura de red utilizada. Para esto utilizamos la métrica f1 sobre el conjunto de test.

Clase	# ejemplos de entrenamiento	# ejemplos de validación	# ejemplos de test
Sweet	401	85	86
Bitter	463	99	99

Resultados

Los clasificadores generados demostraron un rendimiento muy bueno, logrando una exactitud y f1 cercana al 90% en promedio. Contrariamente a nuestra intuición, las arquitecturas convolucionales no resultaron ser ampliamente superiores que la red feedforward. A continuación presentamos las curvas precision-recall y ROC de los modelos más eficaces de cada arquitectura y como se observa en las figuras, el AUC de la graph convolutional obtiene el máximo valor situado en 0.93 mientras que las demás arquitecturas se mantienen muy cercanas.

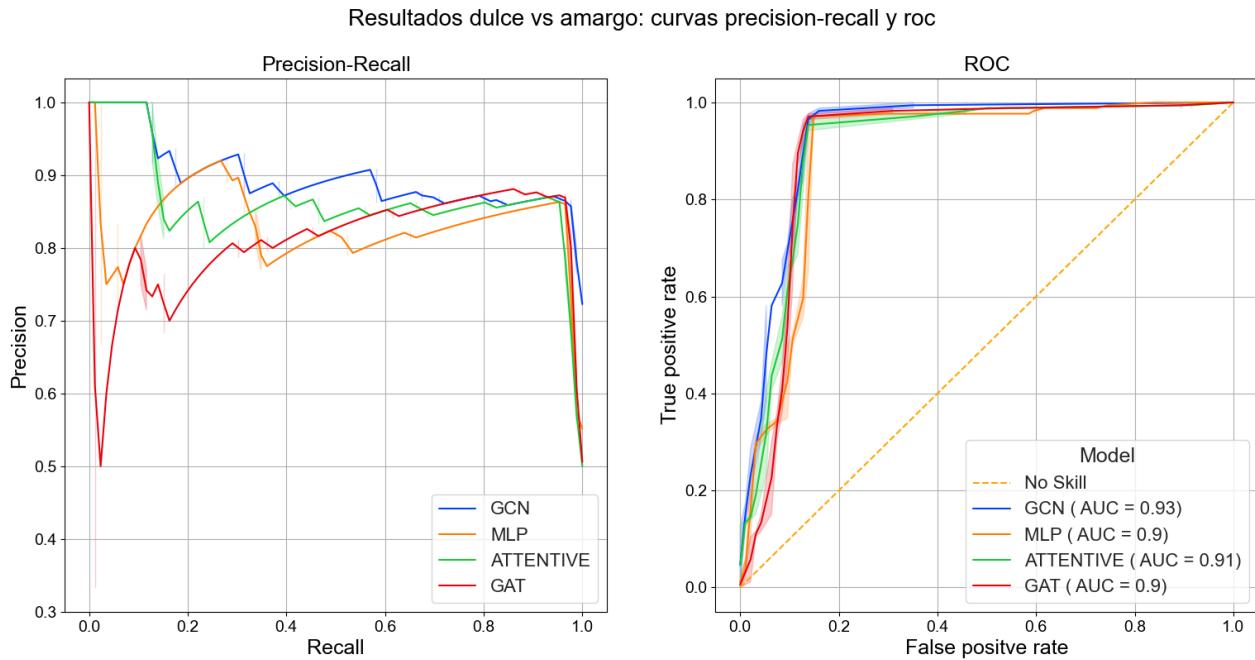


Fig. 5.1: En las figuras izquierda y derecha se observan las curvas de precision-recall y ROC obtenidas con los modelos mas performantes de cada arquitectura aplicadas al dataset de test sweet-bitter. En azul la graph convolutional network, en naranja la red feedforward, en rojo la graph attention network y en verde la red attentive.

Modelo	Exactitud	Precision	Recall	F1
MLP	0.91	0.86	0.97	0.91
GCN	0.92	0.87	0.98	0.92
GAT	0.92	0.87	0.97	0.92
ATTENTIVE	0.91	0.86	0.95	0.90

Luego analizamos la distribución de los scores dados por los modelos para cada clase, esto implica correr el modelo para todas las moléculas del dataset de test sweet y bitter por separado y generar la función de densidad para cada conjunto, de esta forma tenemos una visión más clara de como los modelos separan ambas clases. Los clasificadores fueron capaces de generar distribuciones de probabilidad muy poco superpuestas para ambas clases, en el intervalo ubicado entre 0.75 y 1 puede apreciarse un mayor desvío asociado a la clase 0 (bitter). Finalmente computamos la matriz de confusión de los cuatro modelos.

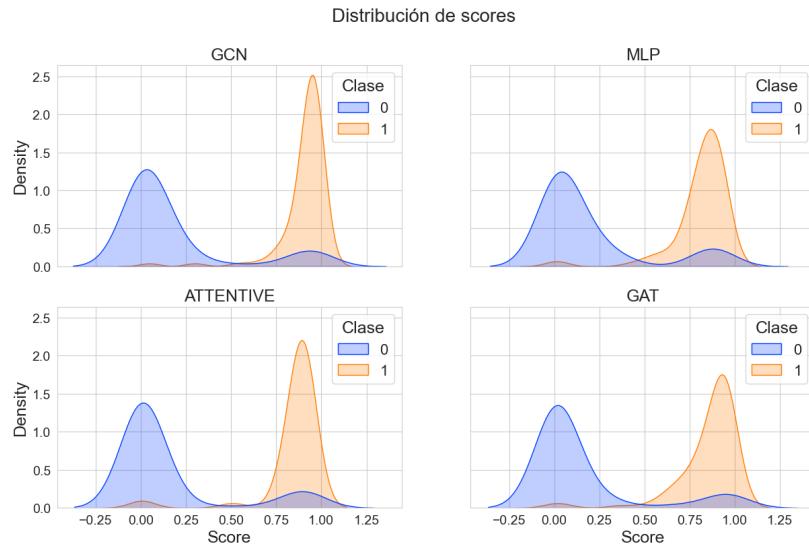


Fig. 5.2: En cada una de las cuatro figuras se puede observar la distribución de probabilidad que dio cada modelo para las moléculas etiquetadas como sweet y bitter. El gráfico en azul(0) se corresponde con la clase “bitter” y el gráfico en amarillo(1) con la clase “sweet”. La intersección entre las distribuciones en cada una de las figuras se corresponde con moléculas que no pudieron ser correctamente discriminadas por los modelos.

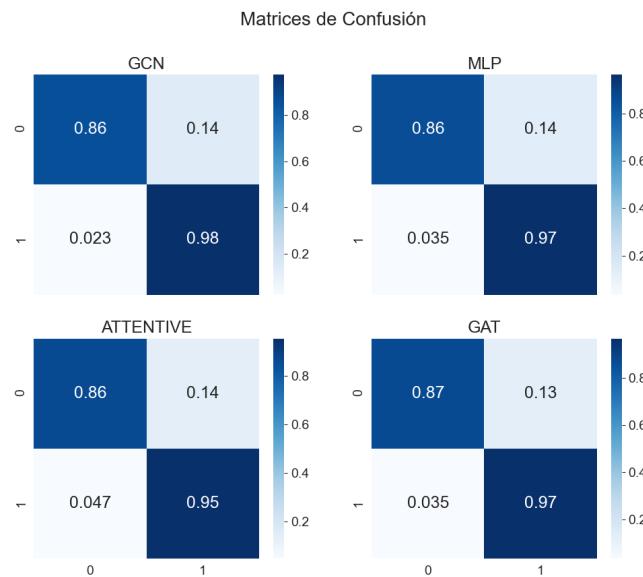


Fig. 5.3: En la figura podemos observar la matriz de confusión de cada modelo. En la matriz correspondiente al modelo convolucional (GCN), el 86 % de las moléculas bitter (clase 0) fueron correctamente clasificadas mientras que el 14 % fueron clasificadas como sweet (clase 1). Mientras que para esta última clase un 98 % de las moléculas fueron clasificadas correctamente y un 2 % de ellas fueron clasificadas como bitter.

Las matrices de confusión ponen en manifiesto lo observado en las distribuciones de scores de las redes, todos los clasificadores cometan un mayor error al clasificar las moléculas amargas como dulces, en cambio el porcentaje de error para la clase dulce es casi nulo.

A continuación se muestran los hiperparámetros de entrenamiento utilizados en cada modelo:

Hiperparámetros										
modelo	l.r.	w.d.	d.o.	b.n.	opt	conv 1	conv 2	f.c #	pool	
MLP	0.01	0	0.1	true	adam	-	-	1	add	
GCN	0.01	0	0	true	adamW	in: 52 out: 96	-	2	add	
GAT	0.015	0	0.1	true	adam	in: 52 out: 64	in: 64 out: 64	1	add	
ATTENTIVE	0.01	0	0.2	true	adam	in: 52 out: 150	-	1	add	

5.3. Multiclasificación de sabores

Preliminares

En este experimento se entrenaron clasificadores multilabel con el fin de detectar la presencia o ausencia de diversas etiquetas asociadas a una molécula. Para esto optamos por un subconjunto de etiquetas específico. Inicialmente comenzamos probando con subconjuntos de dos o tres etiquetas y fuimos agregando nuevas etiquetas a los modelos progresivamente siempre y cuando los resultados obtenidos fueran aceptables. Para esto debemos considerar algunos factores como la distribución no uniforme de las etiquetas (738 bitter vs 540 green p.e.) y la reducida cantidad de etiquetas con un número aceptable de ejemplos para entrenamiento y testing, la sexta etiqueta con más cantidad de apariciones es *woody* con 243 ejemplos. Para este experimento tomamos un 60 % del dataset para entrenamiento y 20 para validación y testeo y realizamos primero cincuenta experimentos sobre el primer espacio de búsqueda de parámetros y luego otros cincuenta experimentos con los parámetros refinados. El conjunto final de etiquetas que optamos por presentar esta dado por *bitter*, *fruity*, *green*, *floral* y *woody* con la siguiente distribución de ejemplos por categoría:

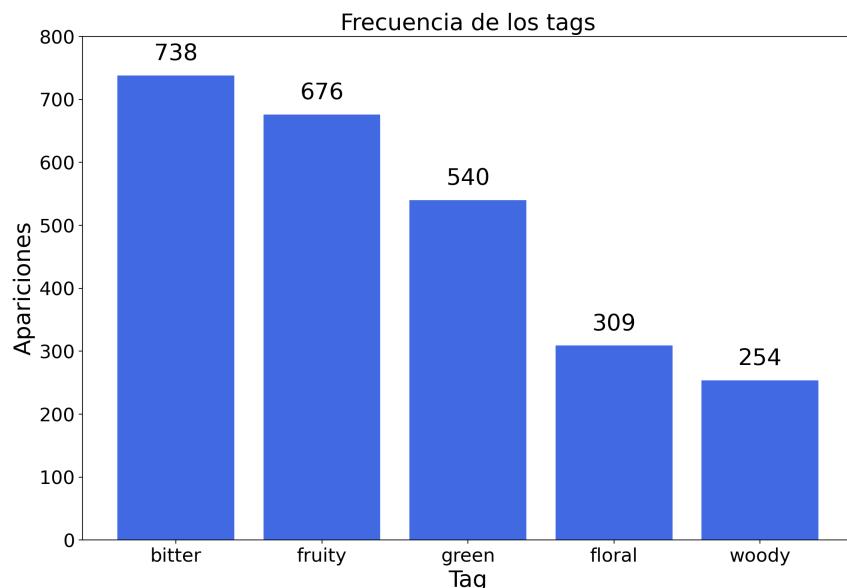


Fig. 5.4: Cantidad de moléculas etiquetadas con los sabores “bitter”, “fruity”, “green”, “floral” y “woody” disponibles para entrenamiento. Se puede observar un claro desbalanceo en la proporción de ejemplos entre clases.

Un problema inherente a la generación de datasets multietiqueta consiste en que cada molécula puede tener más de una etiqueta asociada, por ejemplo, una molécula podría tener un olor maderoso floral dado por las etiquetas “woody” y “floral”. Esto agrega una capa de complejidad extra al generar las particiones de los datos si se busca obtener conjuntos con iguales proporciones para cada categoría en los conjuntos de entrenamiento, testeo y

validación. Por ejemplo, al considerar un ejemplo positivo para la clase maderosa podríamos estar al mismo tiempo agregando un ejemplo de la clase floral o verde, o ninguno de ellos. Para poder mantener las proporciones balanceadas 60/20/20 en cada categoría se implementó un algoritmo utilizando una heurística basada en ir construyendo progresivamente las particiones al agregar primero las clases menos frecuentes a los conjuntos de entrenamiento, validación y testing y sucesivamente ir computando la cantidad de ejemplos faltantes de las clases más numerosas de a una por vez hasta completar las proporciones adecuadas con ejemplos que no rompan el balanceo de las clases ya procesadas. Al ser de naturaleza heurística el algoritmo no puede garantizar siempre que las clases vayan a quedar perfectamente distribuidas pero para el dataset que construimos fue capaz de mantener las proporciones correctas.

Luego de particionar el dataset en los conjuntos de entrenamiento, testeo y validación obtuvimos las siguientes cantidades de ejemplos por categoría:

Partición de entrenamiento		
Etiqueta	Ejemplos positivos (#)	Ejemplos negativos (#)
Bitter	443	759
Fruity	406	796
Green	324	878
Floral	185	1017
Woody	152	1050

Partición de Testing		
Etiqueta	Ejemplos positivos (#)	Ejemplos negativos (#)
Bitter	148	252
Fruity	135	265
Green	108	292
Floral	62	338
Woody	51	349

Partición de Validación		
Etiqueta	Ejemplos positivos (#)	Ejemplos negativos (#)
Bitter	147	264
Fruity	135	276
Green	108	303
Floral	62	349
Woody	51	360

Criterio de selección

Para elegir los mejores modelos asociados a cada arquitectura seleccionamos aquellos con menor perdida en el conjunto de testing. De esta forma evitamos la utilización de métricas específicas y por ende la necesidad de calcular umbrales particulares para cada categoría, los cuales pueden introducir sesgos basados en la aplicación real que podría hacerse de los modelos. Es decir, para ciertos usos podría ser requisito contar con modelos más precisos en ciertas categorías independientemente de tener un bajo recall mientras

que en otros casos podría requerirse un recall más alto al precio de tener modelos menos certeros.

Resultados

Para analizar los resultados computamos las curvas ROC (y su respectivo AUC) y de precision-recall para cada una de las cinco etiquetas y comparamos el rendimiento de cada arquitectura de red agrupando en un mismo gráfico los resultados obtenidos. A continuación vemos las curvas ROC obtenidas para cada etiqueta:

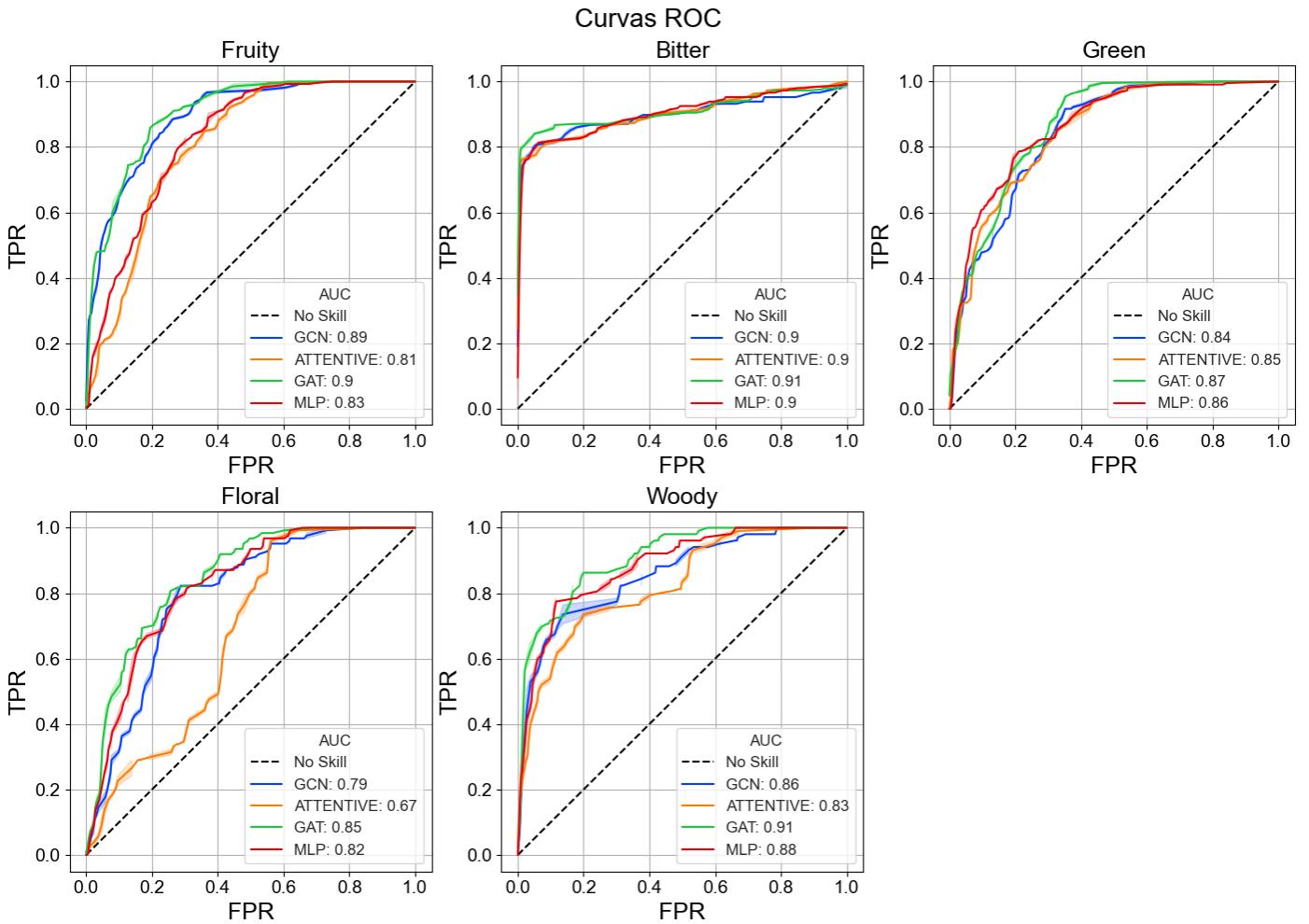


Fig. 5.5: En las cinco figuras superiores se observan las curvas ROC obtenidas para cada una de las cinco clases de sabores con los modelos mas performantes de cada arquitectura aplicados al dataset de test multilabel. En azul la graph convolutional network, en naranja la red feedforward, en rojo la graph attention network y en verde la red attentive. El modelo correspondiente a cada arquitectura es el mismo en cada figura pero aplicado a los distintos labels.

Es destacable que la Graph Attention Network(GAT) domina en todas las etiquetas mientras que la red ATTENTIVE es la que peor rendimiento tiene, por debajo de las otras dos arquitecturas convolucionales y el perceptrón multicapa. Las curvas asociadas a la etiqueta bitter resultan casi idénticas en los cuatro modelos, casualmente es la etiqueta más frecuente y con más ejemplos positivos en el dataset. Por otro lado se puede observar que en la categoría “fruity” se encuentra la mayor diferencia entre alguna arquitectura convolucional y el perceptrón multicapa, con áreas debajo de la curva equivalentes a 0.9(GAT) y 0.83(MLP).

Por último observamos las curvas de precision-recall obtenidas para cada modelo:

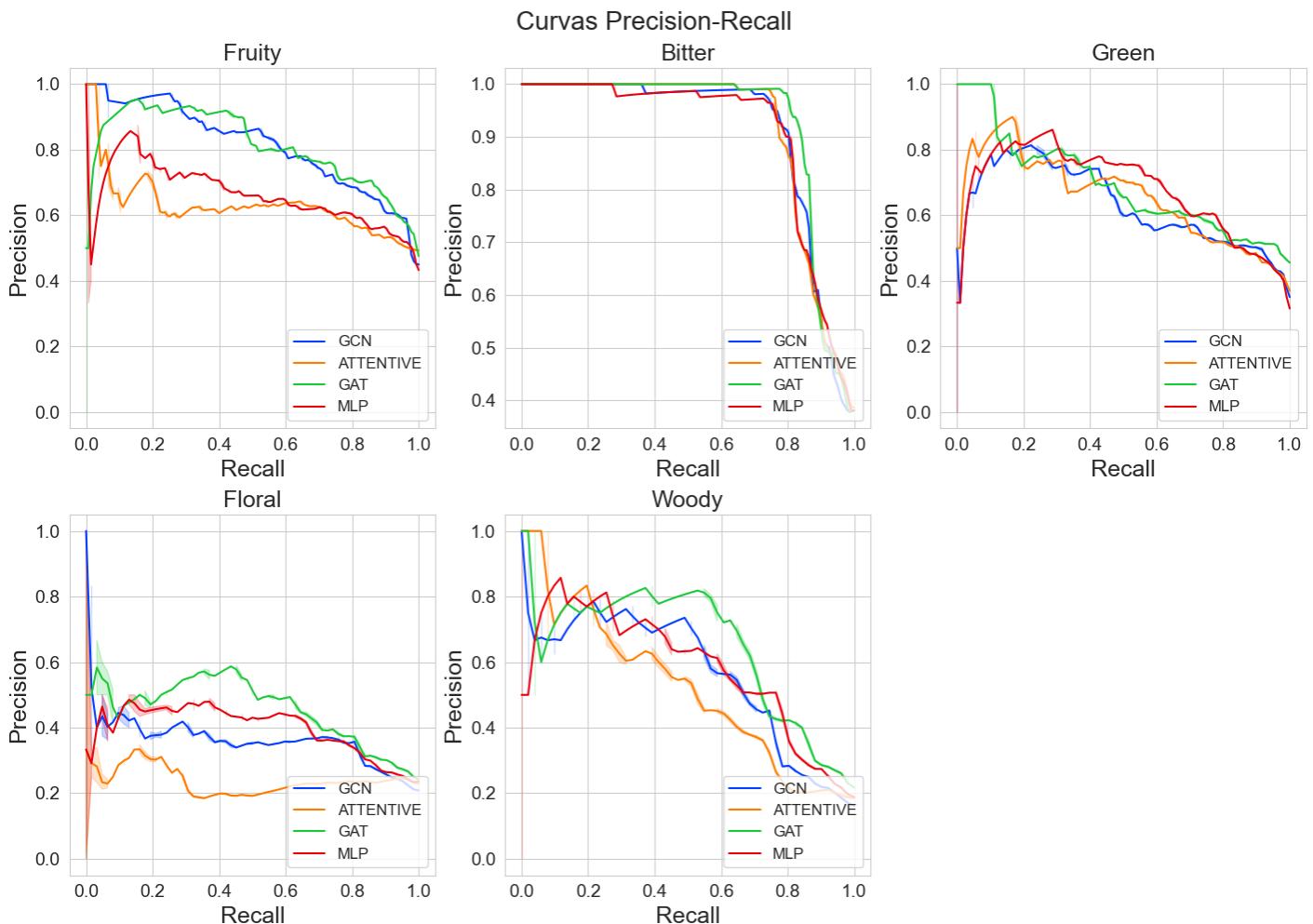


Fig. 5.6: En las cinco figuras superiores se observan las curvas precision-recall obtenidas para cada una de las cinco clases de sabores con los modelos mas performantes de cada arquitectura aplicados al dataset de test multilabel. En azul la graph convolutional network, en naranja la red feedforward, en rojo la graph attention network y en verde la red attentive. El modelo correspondiente a cada arquitectura es el mismo en cada figura pero aplicado a los distintos labels.

Se puede observar un comportamiento análogo a las curvas ROC, además podemos notar que todos los modelos tuvieron mayores inconvenientes para detectar la clase “floral” cuya precisión se encuentra por debajo del 60 % para todos los umbrales. La curva del MLP es levemente superior en la clase “green” y supera o mantiene un comportamiento equiparable a la GCN en las clases “floral”, “woody” y “bitter” mientras que la GAT parece dominar el resultado global y la red ATTENTIVE se mantiene debajo de las demás arquitecturas salvo por un leve intervalo (0.4 - 0.6) en el gráfico de la etiqueta “green” donde es superior a las otras redes convolucionales.

Sobre los modelos

Las redes mantuvieron la misma estructura del experimento anterior compuestas primero por una o más capas convolucionales y seguidas por una o más capas fully connected, a excepción del perceptrón multicapa que no contiene convoluciones. Al ser un problema de clasificación multietiqueta utilizamos cinco neuronas en la capa de salida seguidas por la aplicación de la función sigmoid sobre cada neurona. La función de perdida utilizada fue Binary Cross Entropy With Logistic Loss y en las capas ocultas se utilizó la función de activación *ReLU*.

Los hiperparámetros de entrenamiento usados en cada modelo pueden verse en la siguiente tabla:

Hiperparámetros										
modelo	l.r.	w.d.	d.o.	b.n.	opt	conv 1	conv 2	f.c #	pool	
MLP	0.01	0.01	0.2	true	adamW	-	-	3	max	
GCN	0.02	0.01	0	true	adamW	in: 96 out: 96	-	2	max	
ATTENTIVE	0.01	0.01	0.2	true	adamW	in: 96 out: 16	-	1	add	
GAT	0.01	0.1	0	true	adamW	in: 96 out: 96	-	1	max	

5.4. Detección de moléculas con olor

Preliminares

En este experimento buscamos que la red aprendiera a detectar la presencia o ausencia de olor en las moléculas. Se generaron las particiones correspondientes utilizando la información de las etiquetas presentes en la columna *flavor profile* y de la columna *odor* del dataset. El conjunto de moléculas sin olor se armó teniendo en cuenta aquellas moléculas que presentaban explícitamente la etiqueta “odorless” en el conjunto de etiquetas o bien en su descripción de olor. Por otro lado para el conjunto de moléculas con olor se tuvieron en cuenta solamente aquellas que contuvieran las palabras “strong”, “pungent” u “odor” en su descripción de olor y que no contengan asociada la palabra “odorless” en ninguno de sus atributos de descripción de sabor u olor, de esta forma consideramos principalmente moléculas que tengan un olor fuerte y característico. El dataset generado quedó conformado por 343 moléculas con olor y por 254 sin olor de las cuales tomamos un 60 % para entrenamiento y 20 % para validación y testeo. Se corrieron cien experimentos en el primer espacio de búsqueda y luego cincuenta experimentos con los parámetros refinados y se seleccionaron los modelos más precisos en base a la métrica f1. Se consideró a la clase de moléculas con olor como la positiva (1) y la clase de moléculas sin olor como la negativa (0).

Clase	# ejemplos de entrenamiento	# ejemplos de validación	# ejemplos de test
Con olor	198	72	73
Sin olor	149	47	58

Resultados

Las curvas de precision-recall y ROC muestran que los modelos pudieron separar ambas clases de moléculas, las arquitecturas convolucionales y el perceptrón multicapa tuvieron rendimientos similares, la red ATTENTIVE fue la peor en rendimiento.

Modelo	Exactitud	Precision	Recall	F1
MLP	0.91	0.87	0.97	0.92
GCN	0.90	0.88	0.94	0.91
GAT	0.90	0.92	0.92	0.92
ATTENTIVE	0.74	0.68	0.98	0.80

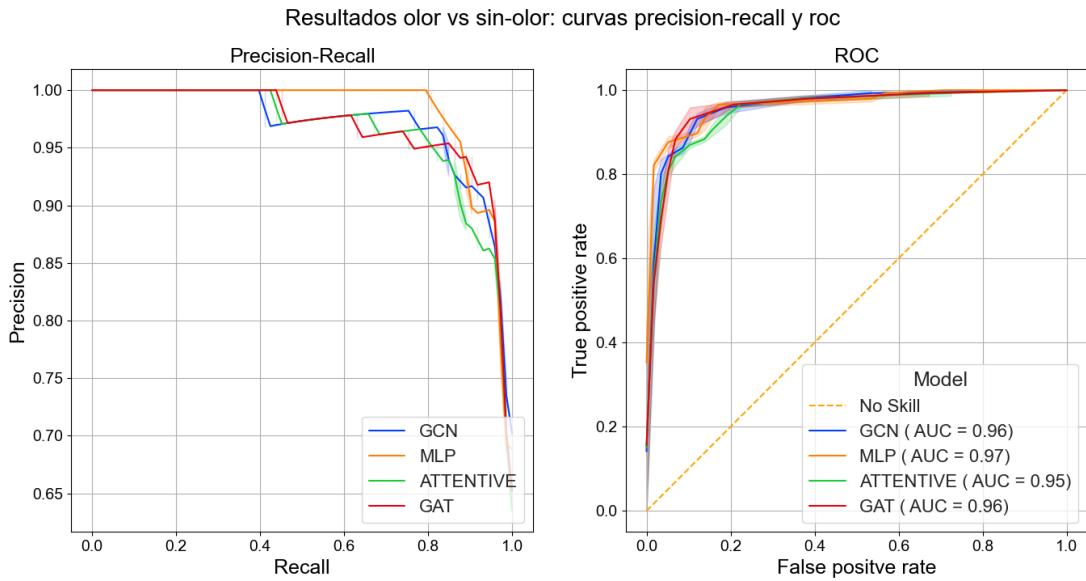


Fig. 5.7: En las figuras izquierda y derecha se observan las curvas de precision-recall y ROC obtenidas con los modelos mas performantes de cada arquitectura aplicadas al dataset de test olor-sin olor. En azul la graph convolutional network, en naranja la red feedforward, en rojo la graph attention network y en verde la red attentive.

La distribución de scores de los modelos muestra intersecciones en las distribuciones de probabilidad en el entorno del 0.5.

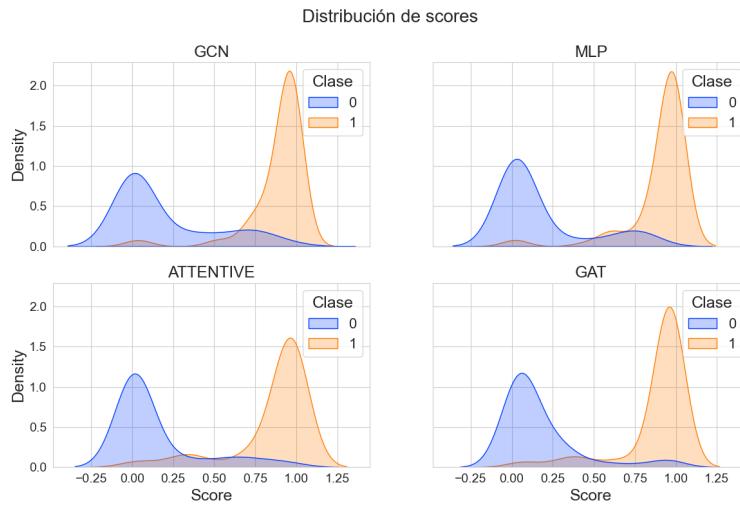


Fig. 5.8: En cada una de las cuatro figuras se puede observar la distribución de probabilidad que dio cada modelo para las moléculas etiquetadas con y sin olor. El gráfico en azul(0) se corresponde con la clase “sin olor” y el gráfico en amarillo(1) con la clase “con olor”. La intersección entre las distribuciones en cada una de las figuras se corresponde con moléculas que no pudieron ser correctamente discriminadas por los modelos.

Los clasificadores cometieron más errores al clasificar equivocadamente las moléculas sin olor como moléculas con olor:

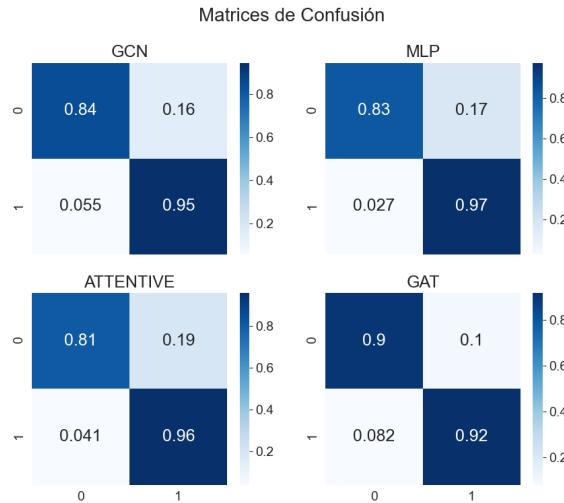


Fig. 5.9: En la figura podemos observar la matriz de confusión de cada modelo. En la matriz correspondiente al modelo convolucional (GCN), el 84 % de las moléculas sin olor (clase 0) fueron correctamente clasificadas mientras que el 16 % fueron clasificadas como olorosas (clase 1). En el grupo de las moléculas con olor un 95 % fueron clasificadas correctamente y un 5 % de ellas fueron clasificadas como sin olor.

Los hiperparámetros de entrenamiento utilizados son los siguientes:

Hiperparametros									
modelo	l.r.	w.d.	d.o.	b.n.	opt	conv 1	conv 2	f.c #	pool
MLP	0.005	0	0.1	true	adam	-	-	1	add
GAT	0.01	0	0.2	true	adamW	in: 49 out: 96	-	3	add
GCN	0.015	0	0.2	true	adamW	in: 49 out: 96	-	3	add
ATTENTIVE	0.005	0	0.2	true	adamW	in: 49 out: 8	-	1	add

6. CONCLUSIONES Y TRABAJO FUTURO

En este trabajo estudiamos el problema del modelado y clasificación de moléculas basados en las propiedades estructurales y lo abordamos mediante la implementación de redes neuronales de grafos, un nuevo enfoque en el mundo del deep learning que viene surgiendo en los últimos años y promete muchos avances en el futuro cercano en los campos relacionados al procesamiento de grafos mediante algoritmos de aprendizaje automático. Vimos como representar digitalmente las moléculas mediante grafos y las bases matemáticas que subyacen en los mecanismos detrás de las convoluciones. Finalmente pusimos en práctica algunas de estas arquitecturas para distinguir propiedades de sabores y olores en el dataset de FlavorDB. Pudimos observar que estas arquitecturas son capaces de igualar y mejorar a los perceptrones multicapa en algunos de los experimentos pero no resultaron ser ampliamente superiores como podría haberse esperado, al menos para los experimentos que aquí fueron planteados. Aun así todos los modelos tuvieron un alto rendimiento en los problemas de clasificación binarios, mientras que en el experimento de clasificación multilabel demostraron complicaciones para distinguir algunas clases de sabores. Uno de los mayores bloqueantes que encontramos tiene que ver con la limitada cantidad de datos con que contamos para entrenar los modelos. Como observamos en la sección 2, a pesar de contener mas de 25.000 moléculas la base de datos, la mayor cantidad de etiquetas se encuentra presente en un número limitado de ejemplos. Otro de los limitantes esta dado por la representación bidimensional de las moléculas ya que la codificación en InChI no permite ubicar tridimensionalmente los átomos en el espacio. Un posible trabajo futuro que extienda esta línea de investigación podría relacionarse con los siguientes puntos:

- Utilizar mejores representaciones moleculares de modo que los modelos puedan capturar con mas exactitud la esencia de la realidad microscópica. Un ejemplo de esto es la utilización de **farmacóforos**, definido por la IUPAC como “un conjunto de rasgos estéricos y electrónicos necesarios para asegurar las óptimas interacciones supramoleculares con un blanco biológico específico y desencadenar (o bloquear) su respuesta biológica”. El uso de farmacóforos podría optimizar la representación de algunas moléculas al brindar una representación mas eficiente de las propiedades que causan la interacción biológica. También podría intentar utilizarse una representación tridimensional de las moléculas con el fin de mejorar la geometría que interpretan los modelos.
- Explorar arquitecturas de redes mas complejas que incluyan el uso en conjunto de otros operadores como por ejemplo las convoluciones clásicas sobre matrices.
- Expandir y mejorar el dataset tomando información de nuevas bases de datos para aumentar la cantidad y calidad de los ejemplos de entrenamiento.
- Implementar los modelos sobre otras clases de etiquetas que no fueron exploradas en este trabajo como “spicy”, “fresh”, “herbal”, etc. y buscar entender si existen clusters de etiquetas relacionadas implícitos en los datos, como por ejemplo podrían ser “green” con “herbal” o “citrus” con “fresh”.

- Aplicar estas arquitecturas para abordar otros tipos de problemas que requieran moléculas con ciertas propiedades como sucede en el descubrimiento de nuevos fármacos, la síntesis de compuestos específicos para ciertas industrias como pueden ser los pesticidas en la agropecuaria o nuevos materiales en la construcción.

Haciendo particular énfasis en este ultimo punto, creemos que el campo de la química médica puede beneficiarse ampliamente de estas tecnologías ya que se encuentra continuamente buscando nuevas drogas y fármacos que actúen de forma específica para combatir trastornos y enfermedades. Como dato de color, en 1991 el químico Alexander Shulgin publicó el trabajo de su vida en el libro *Pihkal* en el cual explora el descubrimiento, síntesis y efectos de 179 feniletilaminas, un grupo de fármacos con propiedades psicodélicas y empatógenas. Muchas de estas moléculas fueron descubiertas y probadas en humanos por primera vez gracias a Shulgin, que se autoadministraba las nuevas drogas en busca de descubrir su potencial farmacológico. Por esos años aún las redes convolucionales en grafos no existían y la inteligencia artificial no era aplicada a este tipo de descubrimientos. Es posible creer que estas tecnologías habrían ayudado a Shulgin en el proceso de exploración, sobre todo a la hora de definir el orden de las moléculas a sintetizar o aportando información sobre los posibles efectos que podrían haber suscitado en si mismo con cierto grado de confianza. Tal vez en un futuro cercano sus investigaciones puedan ser retomadas y sea posible avanzar con mayor velocidad y precisión en el proceso de descubrimiento de moléculas y fármacos con propiedades beneficiosas para el ser humano.

Bibliografía

- [1] <https://www.sciencephoto.com/media/1155563/view/oxygen-atom-illustration>.
- [2] <https://www.toppr.com/guides/chemistry-formulas/hydrogen-sulfate-formula/>.
- [3] https://en.wikipedia.org/wiki/Sulfuric_acid.
- [4] <https://www.msdmanuals.com/es/hogar/trastornos-otorrinolaringolÃ±gicos/sÃ¡ntomas-de-las-enfermedades-de-la-nariz-y-la-garganta/introducciÃ³n-al-olfato-y-al-gusto>.
- [5] <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6627536>.
- [6] https://colab.research.google.com/github/phlippe/uvadlc_notebooks/blob/master/docs/tutorial_notebooks/tutorial7/GNN_overview.ipynb.
- [7] <https://www.ibm.com/cloud/learn/neural-networks>.
- [8] https://cosylab.iiitd.edu.in/flavordb/molecules?common_name=vanillin&functional_group=&flavor_profile=&fema_flavor=&molecular_weight_from=&h_bond_donors=&h_bond_acceptors=&type=&smile=&page=1.
- [9] <https://www.inchi-trust.org/technical-faq-2>.
- [10] <http://www.chemistryland.com/CHM151S/08-Bonds/Bonds/bonds151.html>.
- [11] https://en.wikipedia.org/wiki/Carbon_tetrafluoride#/media/File:Carbon-tetrafluoride-3D-balls-B.png.
- [12] <https://latex-cookbook.net/benzene-ring/>.
- [13] <https://www.quimitube.com/cis-y-trans-dicloroeteno/>.
- [14] https://es.m.wikipedia.org/wiki/Archivo:Axial_chirality_of_spiro_compound.png.
- [15] [https://es.wikipedia.org/wiki/N%C3%BAcleo_\(procesamiento_digital_de_im%C3%A1genes\)](https://es.wikipedia.org/wiki/N%C3%BAcleo_(procesamiento_digital_de_im%C3%A1genes)).
- [16] https://www.researchgate.net/figure/Basic-CNN-block-A-single-layer-is-shown-which-applies-fig5_337463386.
- [17] <https://paperswithcode.com/method/max-pooling>.
- [18] <https://towardsdatascience.com/understanding-graph-convolutional-networks-for-node-classification-101-101>.
- [19] <https://towardsdatascience.com/graph-attention-networks-in-python-975736ac5c0c>.
- [20] <https://relational.fit.cvut.cz/dataset/CORA>.

- [21] Jain D. Patwardhan M. Puri A. Karande S.- Rai B Chacko, R. Data based predictive models for odor perception. *Scientific report*, 2020.
- [22] Richard L Doty, Paul Shaman, Steven L Applebaum, Ronita Giberson, Lenore Sikorski, and Lysa Rosenberg. Smell identification ability: changes with age. *Science*, 226(4681):1441–1443, 1984.
- [23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [24] Minneman KP Hague C, Hall RA. Olfactory receptor localization and function: an emerging role for gpcr heterodimerization. *Mol Interv*, 2004.
- [25] McNaught A. Pletnev I. et al Heller, S.R. *InChI, the IUPAC International Chemical Identifier*. J Cheminform 7, 2015. <https://doi.org/10.1186/s13321-015-0068-4>.
- [26] T Hummel, G Kobal, H Gudziol, and AJEA Mackay-Sim. Normative data for the “sniffin’sticks” including tests of odor identification, odor discrimination, and olfactory thresholds: an upgrade based on a group of more than 3,000 subjects. *European Archives of Oto-Rhino-Laryngology*, 264(3):237–243, 2007.
- [27] Gerkin R. C. Guan Y. Dhurandhar A. Turu-G. Szalai B. Mainland J. D. Ihara Y. Yu C. W. Wolfinger R. Vens C. Schietgat L. De Grave K. Norel R. DREAM Olfaction Prediction Consortium Stolovitzky G. Cecchi G. A. Vosshall L. B. Meyer P Keller, A. Predicting human olfactory perception from chemical features of odor molecules. *Science (New York, N.Y.)*, 2017.
- [28] Jinks AL Hutchinson I Laing DG, Legha PK. Relationship between molecular structure, concentration and odor qualities of oxygenated aliphatic molecules. *Chem Senses*, 2003.
- [29] Zarzo M. Effect of functional group and carbon chain length on the odor detection threshold of aliphatic compounds. *Sensors (Basel)*, 2012.
- [30] Magali Deleu Laurence Lins Marie-Laure Fauconnier Manon Genva, Tierry Kenne Kemene. Is it possible to predict the odor of a molecule on the basis of its structure? *International journal of molecular sciences vol. 20, 12 3018*, 2019.
- [31] Arantxa Casanova Adriana Romero Pietro Lio-Yoshua Bengio Petar Velickovic, Guillem Cucurull. Graph attention networks. *ICLR*, 2018.
- [32] W Pickenhagen. Flavor chemistry—the last 30 years. *Springer US*, 1999.
- [33] Peterlin Z et al Poivet E, Tahirova N. Functional odor classification through a medicinal chemistry approach. *Science Advances*, 2018.
- [34] Rafaeli A. Shaaya, E. Essential oils as biorational insecticides—potency and mode of action. *Springer Berlin Heidelberg*, 2007.
- [35] Agnieszka Sorokowska and Thomas Hummel. Polska wersja testu sniffin’sticks—adaptacja i normalizacja. *Otolaryngologia Polska*, 68(6):308–314, 2014.

- [36] Agnieszka Sorokowska, Valentin Alexander Schriever, V Gudziol, Cornelia Hummel, A Hähner, Emilia Iannilli, Charlotte Sinding, M Aziz, HS Seo, Simona Negoias, et al. Changes of olfactory abilities in relation to age: odor identification in more than 1400 people aged 4 to 80 years. *European archives of oto-rhino-laryngology*, 272(8):1937–1944, 2015.
- [37] Christian Starkenann. Chemistry and biochemistry discoveries in the fragrance and flavor industry. *12th Geneva Chemistry Biochemistry Days (2022) - University of Geneva*, 2022.
- [38] Max Welling Thomas N. Kipf. Semi-supervised classification with graph convolutional networks. *ICLR*, 2017.
- [39] Edwin J. A. Veldhuizen Sara A. Burt Wendy T. Langeveld. Synergy between essential oil components and antibiotics: a review. *Critical Reviews in Microbiology*, 2013.
- [40] Xiaohong Liu Feisheng Zhong Xiaozhe Wan Xutong Li Zhaojun Li Xiaomin Luo Kaixian Chen Hualiang Jiang Zhaoping Xiong, Dingyan Wang and Mingyue Zheng. Pushing the boundaries of molecular representation for drug discovery with graph attention mechanism. *Journal of Medicinal Chemistry*, 2020.