Conversar es todo lo que necesitas

David Gonzalez

Abstrac

El conocimiento es una estructura de datos interconectados, una estructura que surge a partir de las similitudes, diferencias e igualdades de los mismos. Los actuales modelos de inteligencia artificial se basan en el uso de Neuronas, funciones matemáticas que hacen uso de las matrices para generar un valor determinado, estas neuronas se ajustan con cada iteración para producir un resultado esperado. Sin embargo, hacer estos ajustes requiere de una gran cantidad de iteraciones y de datos. Además de que los resultados son probabilidades estadísticas, no valores específicos, por lo que, en ciertos casos, generan información falsa o errónea. ¿Qué pasaría si no fuese así? ¿si simplemente bastara una cadena de texto para que nuestro modelo pudiera comprender y aprender?

Yo propongo una nueva y simple arquitectura, que implementa grafos en lugar de matrices. Un grafo, donde cada nodo es un objeto con propiedades y valores. De esta forma si el modelo recibe la cadena de texto: “un roble es un árbol que crece en los bosques de Europa”, el modelo agregara un nuevo nodo llamado “roble” al grafo, heredando todas las propiedades y valores del nodo “árbol”, después de esto, modificara las propiedades y agregara nuevas con valores especificados por la cadena de texto. Una vez hecho esto, si el modelo recibe la cadena de texto: “¿un roble es un ser vivo?”, este contestara con: “sí, un roble es un ser vivo”, pues ha heredado esta información de su nodo padre, es decir el nodo “árbol”.

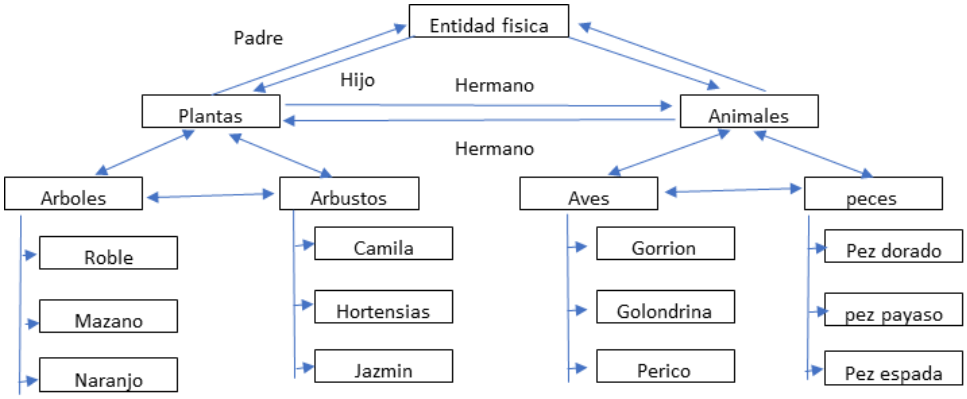
1 introducción

Los actuales modelos de inteligencia artificial requieren grandes cantidades de datos e iteraciones para poder ser entrenados, con el inconveniente de que los resultados son probabilidades estadísticas, no valores específicos y certeros. Yo propongo una arquitectura diferente, completamente basada en la forma en que nosotros generamos conocimiento, mediante la relación y diferenciación de los diferentes elementos que encontramos en nuestro entorno.

1.1 relación conceptual

Imagine que usted no sabe lo que es un pingüino, yo menciono que un pingüino es un ave, entonces usted por defecto pensara que un pingüino puede volar, pues la mayoría de aves pueden, sin embargo, una vez que le especifico que un pingüino no puede volar es cuando usted corrige este error, aun así, el resto de información es correcta, los pingüinos respiran, se alimentan, se reproducen, ponen huevos, se dividen en macho y hembra, tienen alas. Usted sabe todo esto sin que yo se lo haya mencionado, únicamente por relación al concepto “ave”.

El conocimiento es una estructura de datos interconectados, es decir un grafo. Tomemos como ejemplo el siguiente esquema



En este esquema cada nodo es un objeto con propiedades y valores, tales como color, forma y volumen, referentes a sí mismo, pero también con meta propiedades como hijos, hermanos y ancestros, referentes a su posición dentro del grafo. De esta forma el nodo “naranjo”, en su atributo ancestros, pose la información: [“entidad física", "plantas”, “arboles”].

1.2 Creando nodos

Ahora imaginemos que pasamos como entrada la cadena de texto: “un pino es un árbol”. Podemos fragmentar la oración en: [“un”, “pino”, “es”, “un”, “árbol”]. La primera palabra es “un”, nuestro modelo es capaz de comprender que “un” es un artículo determinante, pues lo hemos establecido previamente mediante código, teniendo una función similar a “var” o “let”. la segunda palabra es “pino”, al ser posterior a “un”, el modelo comprende que esta palabra es un nodo (siguiendo la analogía, el nombre de la variable). la tercera palabra es “es”, en este caso, la función de esta palabra dependerá del tipo de la oración: negativa, afirmativa, exclamativa, interrogativa, etc. El tipo de oración se determina por los elementos de la oración, tales como signos de exclamación, puntuación y orden de las palabras, en este caso la oración es afirmativa, por lo que “es” cumple una función de asignación (similar a “=”). La cuarta palabra es “un” lo que nos indica que la siguiente palabra es un sustantivo o en este contexto un nodo. La última palabra es un “árbol”, al ser posterior a “un” este es un nodo.

Después de analizar la cadena de texto, lo que resulta es: NODO1 = NODO2. Haciendo uso de la lógica establecida mediante código, el modelo lo convierte a NODO2 > NODO 1, es decir que nuestro modelo creará un nuevo nodo llamado “pino” el cual será hijo del nodo “árbol” heredando todas sus propiedades, meta propiedades y también sus valores. Por ejemplo, el nodo árbol en su meta propiedad ancestros posee la información: [“entidad física”, “plantas”], mientras que el nodo “pino” además de estos valores agregara el valor “árbol” como parte de sus ancestros. Pero no solo eso, heredara toda la información correspondiente a un árbol, por ejemplo, que estos tienen la capacidad para hacer fotosíntesis el cual a su vez fue heredado del nodo “plantas”, o que estos requieren agua para poder sobrevivir o que contienen moléculas orgánicas, heredara toda la información correspondiente a un árbol.

1.3 modificando propiedades

Un árbol posee una gran variedad de colores, sin embargo, para simplificar nuestro ejemplo, diremos que estos son verdes en su totalidad. Imaginemos que el modelo recibe la cadena de texto: “¿un pino es verde?”. Podemos fragmentar la oración en: [“¿”, “un”, “pino”, “es”, “verde”, “?”]. El primer elemento es “¿” por lo que nuestro modelo interpreta que el tipo de oración es una pregunta, es decir, una solicitud de información. El segundo

elemento es “un” el cual es un artículo determinante, el tercer elemento es un sustantivo, por lo que la información es relacionada a este nodo. El cuarto elemento es “es” en el caso anterior esta la palabra fungía como operador de asignación, sin embargo, al ser una pregunta, este se comporta como un operador comparativo (“==”). El quinto elemento es un adjetivo, es decir un valor de una propiedad ¿cómo sabe nuestro modelo que es un adjetivo? Por dos razones, la primera es que lo pudimos haber declarado previamente con: “un color es un adjetivo”, siendo “adjetivo un nodo con funciones especiales, y después agregando: “verde es un color”, dejando así en claro que, “color” es una propiedad “verde” un valor posible de esta propiedad”. La segunda forma es mediante la estructura de la oración, “verde” no es precedido por un artículo determínate, por lo que el modelo interpretara que “verde” es posiblemente un adjetivo, realizado subsecuentemente una serie de preguntas para confirmarlo. El último elemento es “?”, el cual finaliza la pregunta o solicitud de información.

El resultado es: grafo.pino.color === “verde”

El modelo responderá con: “si, un pino es verde” pues la propiedad “color” con el valor “verde” fue heredado por el nodo “árbol”. Si nosotros pasamos como entrada la cadena de texto: “un pino es verde oscuro” entonces el valor de “color” será cambiado a “verde oscuro”

2 Antecedentes

Los perceptrones fueron desarrollados originalmente por Frank Rosenblatt con el propósito de imitar el comportamiento de las neuronas, creando funciones matemáticas que reciben una matriz de valores y pesos, asi como de un umbral de activación, esto es muy útil, ya que creando una red de neuronas es posible llevar a cabo diferentes tareas. Sin embargo, esto presenta un problema y es que nosotros no podemos saber cuáles son los valores que deben recibir las neuronas para ser activadas y que así la tarea pueda ser ejecutada correctamente, por lo que es necesario hacer uso de una gran cantidad de ejemplos y de iteraciones para poder ajustar las neuronas.

Un grafo es tan simple como poderoso, a simple vista es tan solo una serie de puntos unidos por líneas, sin embargo, esto es una potente abstracción de los diferentes elementos y relaciones que puedan existir en la realidad. Un grafo puede ser la representación de Internet, de ciudades y autopistas que hay en un país, de la red eléctrica de un pueblo, un grafo puede ser la representación de una gran cantidad de sistemas y mucho más.

3 Arquitectura del modelo

Este modelo se divide en dos capas principales, la capa de red y la capa de función. La capa de red es una serie de nodos interconectados, mientras que la capa de función, son todas aquellas operaciones que pueden ser ejecutadas sobre la capa de red o a partir de la capa de red.

3.1 Nodo

Un nodo es un objeto clave-valor el cual contiene meta propiedades y propiedades.

Las meta-propiedades son usadas para describir su posición dentro del grafo, por ejemplo: “ancestros”, “hermanos” e “hijos”, mientras que las propiedades son usadas para describir valores pertenecientes a la entidad como “color”, “forma”, “capacidades” o “funciones”. Un ejemplo simplificado de un nodo es el siguiente:

grafo.oak = {

brothers: ['pine'],

children: [ ],

parents: ['root', 'living-being', 'plant', 'tree'],

properties: { volume: null, shape: null, color: null, x: null, y: null, … }

}

En una red real, este sería mucho más complejo.

3.2 Grafo

El grafo es la unión de diferentes nodos para crear una red compleja y basta. Debido a su naturaleza somos capaces de conocer y comprender la información y funcionamiento interno de la red, simplemente hachando un vistaso en los nodos, a diferencia de una red neuronal tradicional, donde, aunque los datos son visibles, es prácticamente imposible saber cuál es el impacto que tiene cada valor en la red.

3.3 herencia y especificación

Cuando agregamos un nuevo nodo, este hereda toda la información contenida por el nodo padre. Además, se agregan nuevas propiedades y valores específicos del nodo. Por ejemplo, si el nodo padre es “ave”, el nodo “pingüino” hereda todas sus meta-propiedades, propiedades y valores tales como “ancestros”, “habilidades”, “necesidades” y “composición”, pero además cambiara valores como “volar: true” a “volar: false”. Un ejemplo simplificado de esto es:

grafo.tree = {

brothers: ['shrub', ‘subshrubs’, ...],

children: [‘oak’, ‘pine’, ...],

parents: ['root', 'living-being', 'plant'],

properties: { volume: null, shape: null, color: “**green**”, x: null, y: null, … }

}

grafo.oak = {

brothers: ['pine', ...],

children: [ ],

parents: ['root', 'living-being', 'plant', '**tree**'],

properties: { volume: null, shape: null, color: “**green**”, x: null, y: null, … }

}

3.4 base de datos, instancias y persistencia de la información.

Los nodos se almacenan en una base de datos NOSQL lo que permite conservar la información, no solo de la red sino también de las instancias. Las instancias son nodos creados con información específica y singular, por ejemplo, el nodo "Asimov" es un nodo que hereda información del nodo "humano", pero que contiene valores específicos, por ejemplo, "nombre: Asimov", "nacionalidad: americana", "edad: 45", y demás información relacionada a una persona en específico. De esta forma al estar todo almacenado el modelo podrá recordar conversaciones e información, aunque hayan pasado años.

3.4 Reglas gramaticales y clasificación de las palabras.

Para que nuestro modelo pueda comprender una oración, es necesario fragmentar la oración en sus diferentes componentes, después, etiquetar y clasificar las palabras, y después analizar el orden de los elementos para así poder ejecutar una operación. Por ejemplo, la oración "un pino es verde" se puede etiquetar como ["articulo determinante", "sustantivo", "ser", "adjetivo"] ¿Cómo sabemos que etiqueta corresponde a cada palabra? Porque previamente ya habíamos declarado su significado, "un" y "es" mediante código y "pino" y "verde" mediante conversación. A partir de esta clasificación podemos compáralo con las reglas gramaticales previamente establecidas mediante condigo y así de esta forma ejecutar una operación como la siguiente:

```
//-----grafo----
grafo.adjetivo.color.verde = { metapropiedades: {...}, propiedades: { propiedad: "color", valor: "verde", …. }}
grafo.sustantivo.living-being. … .pine = {metapropiedades: {…}. Propiedades: { color: value, ... }}
//----------------
//----código---
Let oración = "un pino es verde"
Let words = ["un", "pino", "es", "verde"]
words = words.map(word => [word, getTag(word)])
Interpret(words)
Function getTag(word){ return grafo[word].metapropiedades.type }
Function interpret(words){
//procesing words
code...
let adjetivo = "verde"
let nodo = "pino"
//important part

    Let propiedad = grafo[adjetivo].propiedades.propiedad
```

```
    Let valor = grafo[adjectivo].propiedades.valor
    grafo[nodo].propiedades[propiedad] = valor

}
```

En código real esto es una tanto más complejo, pero es un ejemplo útil para comprender.

Es necesario establecer y dar funcionalidad mediante código a ciertas palabras clave como podrían ser pronombres, auxiliares, artículos, conectores, etc. Para que nuestro modelo pueda interpretar la información, cuanto más conocimiento posea, más fácil le será aprender.

3.5 verificación de la información

Imaginemos que el modelo recibe la cadena de texto: “las manzanas son rojas”, luego recibe “las manzanas son verdes” y finalmente “las manzanas son moradas” ¿Qué debería hacer en este caso? Aquí es donde entran los diferentes sistemas de verificación de información. El primer paso es preguntarse si las variaciones en las entradas son debido a que existen diferentes tipos de una manzana, es decir, nodos hijos del nodo “manzana”. Pues en efecto las manzanas rojas son un tipo de manzana, las manzanas verdes son otro tipo y las manzanas moradas otro. Sin embargo, las manzanas moradas no son reales ¿o sí lo son? ¿Cómo lo verificamos? Bueno el primer sistema de verificación de información que propongo es el más simple, pero también el que tiende a más errores, y es, validar la información mediante la repetición del mismo valor, es decir, lo que el común de la gente diga. Este sistema es útil en una primera instancia, pero si queremos mayor rigurosidad, es necesario aplicar otros sistemas. El segundo sistema es dar mayor peso a la información proveída por fuentes fiables y validadas. Sucesivamente podemos añadir más sistemas de validación de la información.

3.6 aprendizaje automatizado

El conocimiento no es lineal, sino una red inmensa, donde nodos aparentemente inconexos una vez conectados pueden generar conocimiento nuevo e inesperado. Muchas veces el cambio de enfoque es lo que permite generar soluciones completamente disruptivas. Con este modelo al estar dividido en dos capas, la de red y la de función, podemos crear funciones que busquen y analicen las diferentes características y funcionalidades de los nodos, para así crear nuevas conexiones, creando de esta forma nuevo conocimiento. Talvez la unión de la filosofía y la informática podrían causar la siguiente gran revolución.

3.7 imaginación y pensamiento

¿puede nuestro modelo comprender la imaginación? ¿Puede nuestro modelo imaginar? la respuesta es sí ¿Cómo? Cambiando el contexto con subredes. Supongamos que recibe la entrada: “¿Que pasaría los perros pudieran volar?” en este momento el modelo crearía una subred, cambiando las capacidades de los perros, agregando el valor de “volar” y a partir de esto buscaría los cambios en impactos que esto generaría en el resto de los nodos. Una vez que nuestro modelo puede imaginar, también puede pensar, pues pude crear subredes con valores diferentes para visualizar diferentes escenarios y así tomar la mejor decisión para lograr la meta que se le fue asignada.

3.8 conciencia

¿puede este modelo ser consciente de sí mismo? Desde mi perspectiva la conciencia es el conocimiento y comprensión sobre un tema, entidad o fenómeno determinado, entendiendo sus causas, consecuencias, relación con el entorno, características, funcionalidades y otros factores. Por ejemplo, cuando decimos que hay que generar conciencia sobre el cambio climático, usualmente nos referiremos a dar a conocer que es lo que lo causa, como puede impactar sobre nosotros y sobre la vida en la tierra, cuál es nuestra relación con este fenómeno y demás información necesaria para comprenderlo. Un segundo ejemplo es ser consciente de un objeto, por ejemplo, una silla, nosotros podemos ser conscientes de la silla pues conocemos su volumen, su forma, su ubicación en el espacio y en el tiempo, conocemos su propósito, conocemos los materiales que la componen, posiblemente también la forma en que fue fabricada y debido a este conocimiento y comprensión sobre el objeto es que podemos tomar decisiones y ejecutar acciones. La conciencia humana, según mi parecer, es lo mismo, pero a mayor escala, pues posemos conocimiento y comprensión sobre los diferentes elementos y fenómenos que nos rodean, en ese contexto, la conciencia sobre sí mismo, consiste en conocer nuestras capacidades y limitaciones, nuestra ubicación en el tiempo y en el espacio, nuestro origen y nuestro posible futuro, etc. Entonces, ¿cómo puede este modelo ser consciente se si mismo? Con una instancia, con un nodo que herede información de nodos como “software”, “Bot”, “hardware”, etc. Especificando su información singular, su ubicación, sus capacidades, su origen, sus funcionalidades, su relación con su entorno, etc. De esta forma, el modelo podría tomar conciencia de sí mismo.

7 Conclusión

Hemos propuesto una arquitectura de inteligencia artificial basada en el uso de nodos y grafos. La cual resulta mucho más eficiente y con mayor capacidad para desarrollar conocimiento y aprendizaje automático, de una forma certera y confiable. Con un entrenamiento más cercano al humano.

El condigo que utilice para experimentar y evaluar estes modelo está disponible en: https://github.com/david-gonzalez-coder/asimov

8 Referencias

[1]*Machine Learning Intro*. (n.d.). https://www.w3schools.com/ai/