



INTELIGENCIA ARTIFICIAL

INTERACCIÓN HUMANO-MÁQUINA



Structuralia

Este documento es de uso único e intransferible para el alumno matriculado en el curso. Cualquier reproducción física o digital del documento sin permiso de los autores vulnera los derechos de propiedad intelectual de los mismos.



INDICE

INDICE	2
1. INTRODUCCIÓN.....	<i>¡ERROR! MARCADOR NO DEFINIDO.</i>
2. LA VISIÓN ARTIFICIAL	4
3. EL PROCESO DE COMPUTER VISION.....	6
3.1 Datasets.....	8
3.2 Filtros	10
3.3 Servicios de computer vision.....	13
4.TAREAS DE APRENDIZAJE EN COMPUTER VISION	17
4.1 El proceso de reconocimiento de una imagen	17
4.2 Las redes de neuronas convolucionales	19
5. EL RECONOCIMIENTO DEL HABLA.....	24
5.1 Los principales elementos del Natural Laguage Processing	25
6. EL PROCESO DE RECONOCIMIENTO DEL LENGUAJE	25
6.1 Analizando el sonido del habla	27
6.2 Text-mining.....	28
6.3 Métricas	29
6.4 Sentimiento del texto	30
6.5 Diferencias entre text-mining y natural language processing	31
7. EL PROCESAMIENTO DEL LENGUAJE NATURAL.....	31
6.1 Servicios de NLP.....	32
6.2 Redes de neuronas recurrentes.....	34
6.3 Redes LSTM.....	39
6.4 Redes LSTM en combinación con redes convolucionales	42

1. INTRODUCCIÓN

Si analizamos cómo se relaciona el ser humano nos daremos cuenta de que principalmente utiliza los ojos y oídos para percibir el entorno y el habla para comunicarse. El sentido de la vista es uno de los sentidos críticos y más importantes ya que permite a los humanos entender e interactuar rápidamente con el entorno que los rodea. Lo mismo sucede con los animales, entre otras cosas les permite detectar potenciales peligros. En el campo de los sistemas inteligentes, la visión artificial nos permitirá analizar qué está sucediendo en el entorno y actuar en consecuencia. En la primera parte de este tema se describirá en detalle qué es la visión artificial y cómo podemos implementarla dentro de un sistema inteligente. Por otro lado, La comunicación es una habilidad esencial en los seres humanos, ya que nos permite entender e interactuar rápidamente con otros humanos. El idioma, la entonación, la semántica y la sintaxis son componentes imprescindibles que debemos analizar a la hora de desarrollar esta capacidad en los agentes inteligentes (chatbots, robots, asistentes...). En la segunda parte de este tema se describirá en detalle qué es el procesamiento del lenguaje natural, el reconocimiento del lenguaje y cómo podemos implementar ambos en un sistema inteligente.

2. LA VISIÓN ARTIFICIAL

El campo de la visión artificial es transversal a muchas áreas tecnológicas y científicas. En la siguiente figura se muestra un ejemplo de clasificación. Dependiendo de en qué área nos encontramos, la visión artificial toma un nombre distinto. El campo de la visión artificial aplicada a la inteligencia artificial se encuentra entre las áreas de Artificial Intelligence y Machine Learning. Esta disciplina se conoce como Computer Vision.

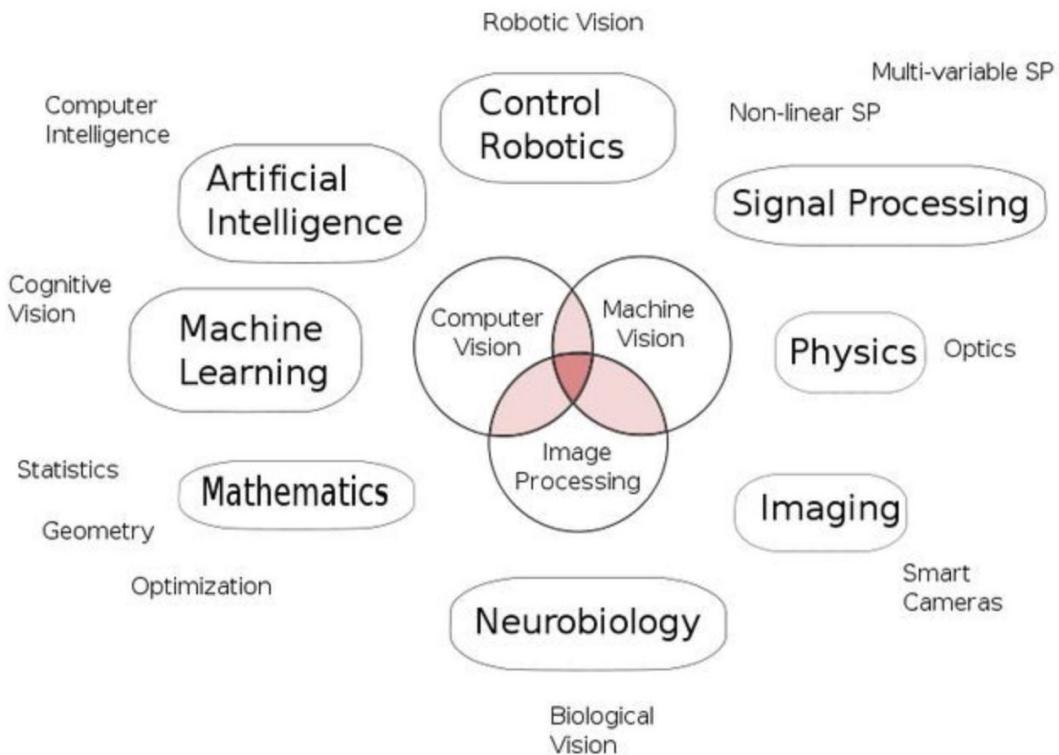


Ilustración 1: Clasificación de la visión artificial

Utilizamos el término Computer Vision para referirnos a cualquier proceso que incluya un procesado de imagen o video offline o en tiempo real (online) para posteriormente realizar tareas de reconocimiento o identificación de elementos sobre el mismo.

2.1. COMPUTER VISION

Computer Vision está directamente relacionado con tareas de extracción de conocimiento, análisis y sintetización de información “útil” obtenidas a través de una imagen o una secuencia de ellas. Para realizar estas tareas se necesita desarrollar una serie de algoritmos sobre los que iremos profundizando en las próximas secciones. Esta área se aplica a cualquier tipo de sector. Opcionalmente puede incluirse un robot para disponer de un cuerpo físico. Lo imprescindible para trabajar en Computer Vision es contar con un conjunto de imágenes.

Ejemplos de tareas de Computer Vision en distintos campos son la agricultura (evaluación del cultivo para su recogida), vehículo autónomo (análisis del entorno en tiempo real), sistemas

biométricos (sacar los puntos clave de una cara para análisis de expresiones o identificación de personas), restauración de imágenes (aplicación de filtros sobre imágenes), seguridad y vigilancia (monitorización para control de situaciones anómalas o de peligro), análisis de la contaminación, de imágenes médicas, de velocidad en carreteras etc.

En concreto, dentro del campo de la robótica, las principales tareas donde se trabaja con técnicas de Computer Vision son:

- Reconocimiento de patrones
- Extracción de características
- Mapping (identificar las zonas navegables)
- Reconocimiento de objetos, personas, escenarios etc.
- Interactuar con humanos

3. EL PROCESO DE COMPUTER VISION

El proceso de Computer Vision está dividido en seis etapas principales. Si se desea desarrollar un sistema robusto habrá que trabajar sobre las seis etapas en conjunto. Si se desea sólo abordar un problema en concreto es posible que no sea necesario trabajar en todas ellas. La siguiente figura muestra el ciclo completo de las seis etapas partiendo de la percepción realizada sobre el entorno y sus propias restricciones (si hay alguna).

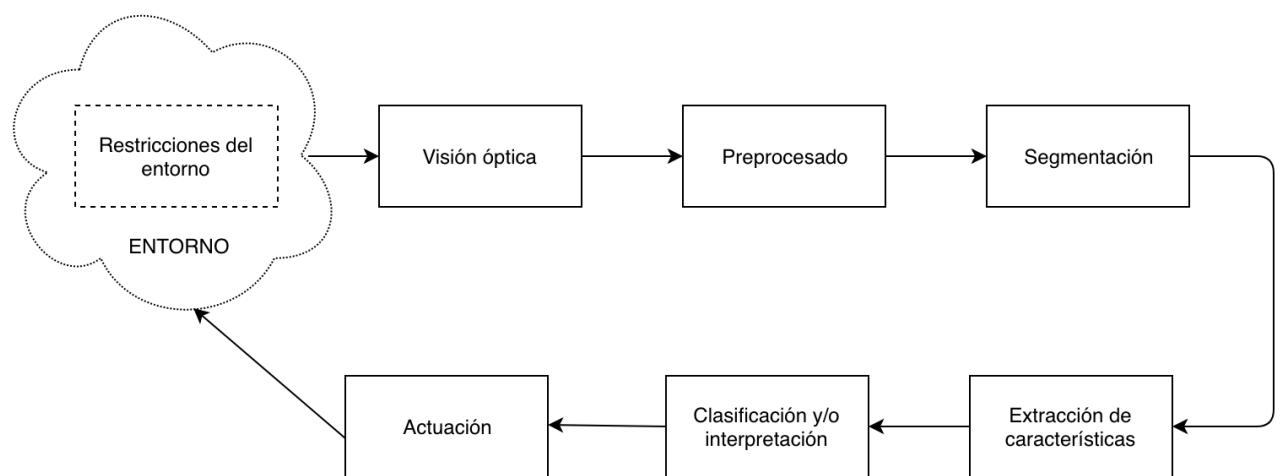


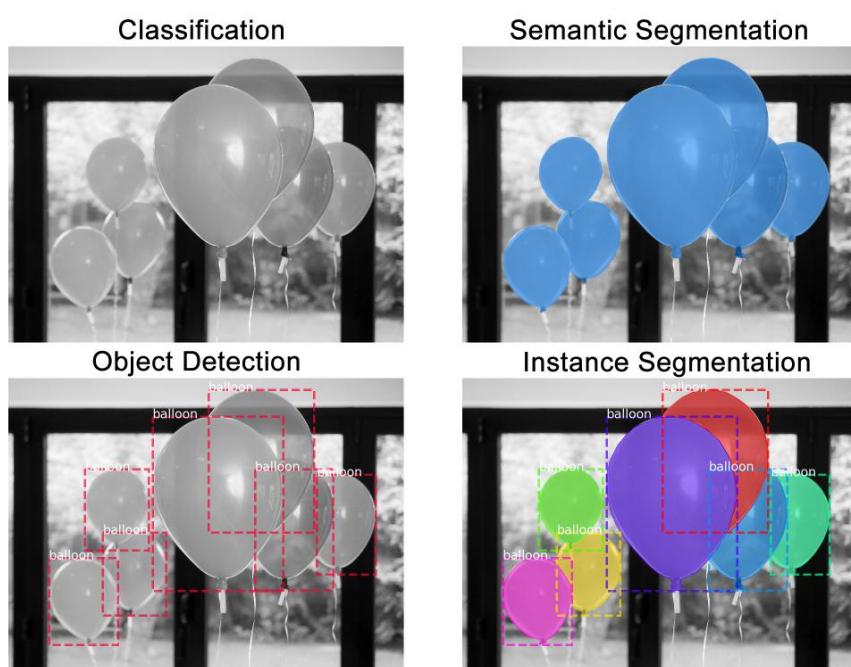
Ilustración 2: El proceso de Computer Vision

El primer paso es realizar un análisis previo del entorno en términos de viabilidad y restricciones. ¿Qué elementos queremos detectar o identificar? ¿Qué sistemas de visión y sensores podemos incorporar? ¿Hay luz natural o artificial? Etc.

Una vez hemos establecido la forma de capturar imágenes o video y tenemos el entorno sensorizado comienza la fase de captura de información. Se recopilarán imágenes y videos del entorno en los momentos del día que nos interese analizar. Si es un entorno cerrado o controlado se intentará replicar el entorno real final con tanta fiabilidad como sea posible.

Ahora que ya se dispone de un conjunto grande y variado de datos (imágenes), se pasa a la fase de preprocesado. La principal tarea del preprocesado es limpiar el dataset o conjunto de datos de ruido (imágenes no representativas, mal tomadas, erróneas...) así como normalizar medidas o utilizar 1 (blanco/negro) o 3 canales (RGB – Rojo Verde Azul).

El siguiente paso se llama segmentación y es una de las claves principales de todo el proceso de Computer Vision. La segmentación nos permite analizar la imagen mediante transformaciones de sus píxeles. Cualquier imagen se representa como una matriz numérica en la que cada elemento numérico representa un píxel de la imagen. De esta forma y mediante la aplicación de filtros conseguiremos transformar la imagen para detectar formas, elementos, colores concretos etc que nos acerquen más a la extracción de características. En la siguiente figura se pueden ver cuatro tipos de segmentación:



Finalmente, una vez se han aplicado diferentes filtros a la imagen, el algoritmo será capaz de extraer diferentes características sobre la misma. Así podrá clasificar la imagen adecuadamente o identificar qué es lo que está observando y actuar en consecuencia. Una vez el algoritmo dispone de esa información, procederá a ejecutar la acción correspondiente sobre el entorno real.

3.1. Datasets

Como se ha comentado previamente, para poder trabajar en Computer Vision se necesitan imágenes. En concreto, una amplia variedad y cantidad de imágenes especialmente si queremos que nuestro sistema sea capaz de aprender a través de las mismas. Un dataset es un conjunto de datos específico para entrenar un algoritmo. En los últimos años, diferentes investigadores se han dedicado a la elaboración de datasets de “auto-tagging o labeling” de imágenes. El auto-tagging o auto-labelling consiste en describir con palabras qué elemento hay en la imagen (tarea básica) y describir qué sucede (tarea compleja). Dos de los datasets más conocidos relacionados con este tipo de tareas son CIFAR-10 y CIFAR-100 [1].

CIFAR-10

El dataset CIFAR-10 consta de 60000 imágenes en color de 32x32 en 10 clases, con 6000 imágenes por clase. Hay 50000 imágenes de entrenamiento y 10000 imágenes de test.

El conjunto de datos se divide en cinco lotes (batch) de entrenamiento y un lote de test, cada uno con 10000 imágenes. El lote de test contiene exactamente 1000 imágenes seleccionadas al azar de cada clase. Los lotes de entrenamiento contienen las imágenes restantes en orden aleatorio, pero algunos lotes de entrenamiento pueden contener más imágenes de una clase que de otra. Entre ellos, los lotes de entrenamiento contienen exactamente 5000 imágenes de cada clase.

CIFAR-100

Este conjunto de datos es como el CIFAR-10, excepto que tiene 100 clases con 600 imágenes cada una. Hay 500 imágenes de entrenamiento y 100 imágenes de test por clase. Las 100 clases en el CIFAR-100 se agrupan en 20 superclases. Cada imagen viene asignada a una primera etiqueta que indica la clase a la que pertenece y una segunda etiqueta que indica la superclass.

En la siguiente referencia podrás encontrar más información sobre los datasets y las clases en las que clasifica las imágenes.

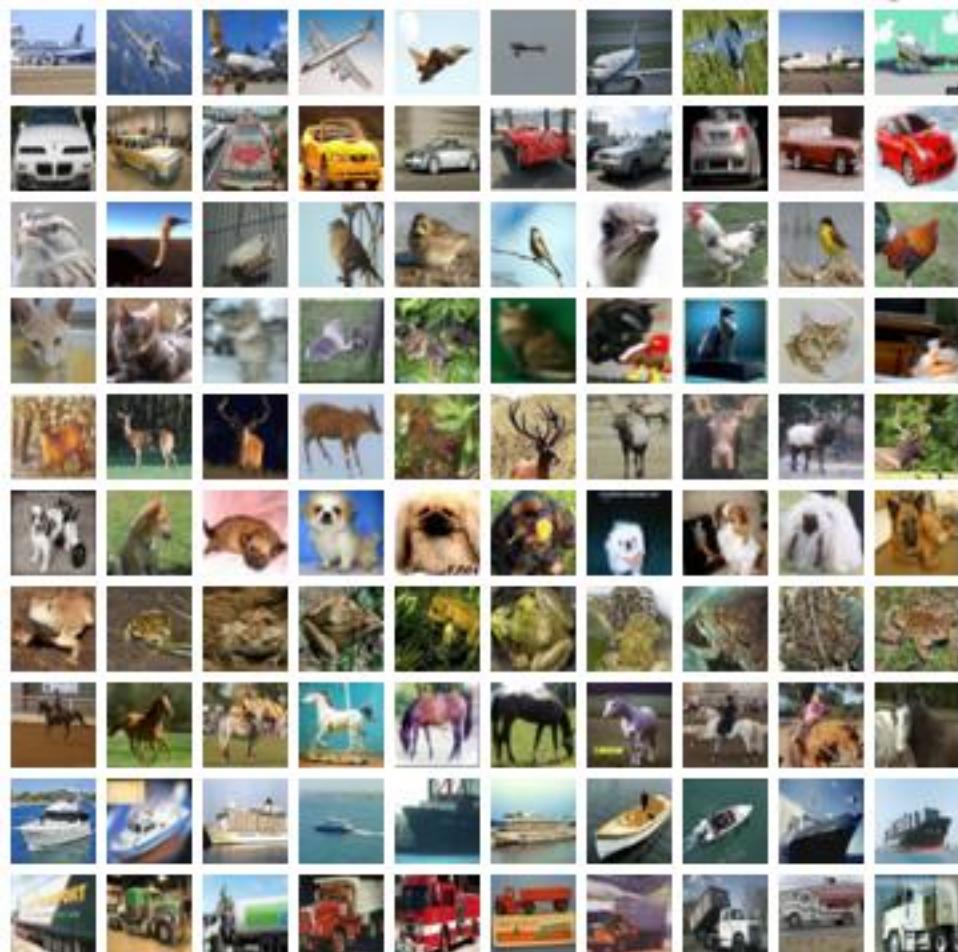


Ilustración 3: Muestra de imágenes de CIFAR

IMAGENET

ImageNet [2] es una base de datos de imágenes organizada según la jerarquía de WordNet en la que cada nodo de la jerarquía está representado por cientos y miles de imágenes. Actualmente contiene un promedio de más de quinientas imágenes por nodo.



Ilustración 4: Muestra de imágenes del dataset IMAGENET

Existen otros datasets que se utilizan en Computer Vision y en Robótica como CINIC-10 o DEX-NET. Los que se han mencionado son los más populares.

Por otro lado, los lenguajes de programación más populares para trabajar en Computer Vision son C++ y Python. En concreto, durante muchísimos años se ha utilizado la librería de referencia OpenCV.

3.2. Filtros

Una imagen se puede traducir a una matriz numérica de la siguiente forma:

$F(x,y)$ representa la intensidad en la posición x,y de la imagen

Si la imagen está representada en blanco y negro será suficiente con utilizar una matriz. Si la imagen está representada en color, será necesario trabajar con tres matrices, una para cada canal (rojo, verde, azul – RGB).

$$f(x,y) = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix}$$

El resultado de esta matriz numérica puede verse en forma de imágenes en la siguiente figura. Cuando se entremezclan los canales azul, verde y rojo obtendríamos la imagen final a color. Cada canal de forma individual representa la intensidad de dicho color en cada píxel de la imagen.

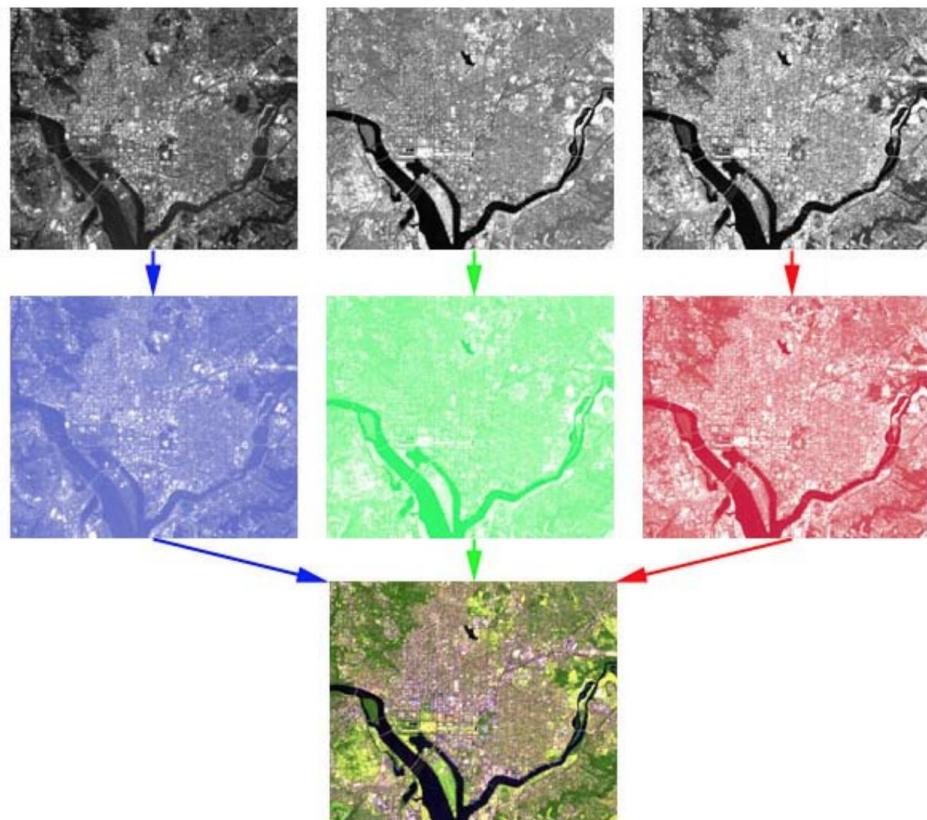


Ilustración 5: Canales RGB de una imagen

Ahora que ya sabemos cómo se convierte una imagen en datos numéricos y los tipos de matrices que podemos encontrarnos se van a presentar una serie de filtros que nos ayudan a realizar determinados elementos de una imagen.

Existen múltiples filtros que aplicar a una imagen. Un filtro es una transformación lineal o no lineal aplicada a parte o toda la imagen. Si el filtro no es convolucional, la salida será una matriz numérica nueva de las mismas dimensiones que la matriz inicial. Por ello, se explican a continuación los dos grandes grupos de filtros no convolucionales:

- Filtros para alta frecuencia: este tipo de filtros aplica una función matemática sobre la matriz que acentúa las partes de mayor contraste de la imagen, es decir, generalmente acentúa los bordes y formas de los elementos que aparezcan en la misma. El filtro más famoso es el de paso alto.



- Filtros para baja frecuencia: este tipo de filtros aplica una función matemática sobre la matriz que suaviza las partes de mayor contraste de la imagen, es decir, generalmente se utiliza para eliminar ruido o disminuir las zonas de mayor intensidad. Este tipo de filtros se conocen también con el nombre de smoothing (suavizado) y el filtro más popular es el gaussiano.



En resumen, para una misma imagen podríamos aplicar estos dos tipos de transformaciones para o bien detectar formas o bien reducir o limpiar el ruido. En la siguiente figura se puede ver el efecto que tienen ambos filtros sobre una misma imagen.

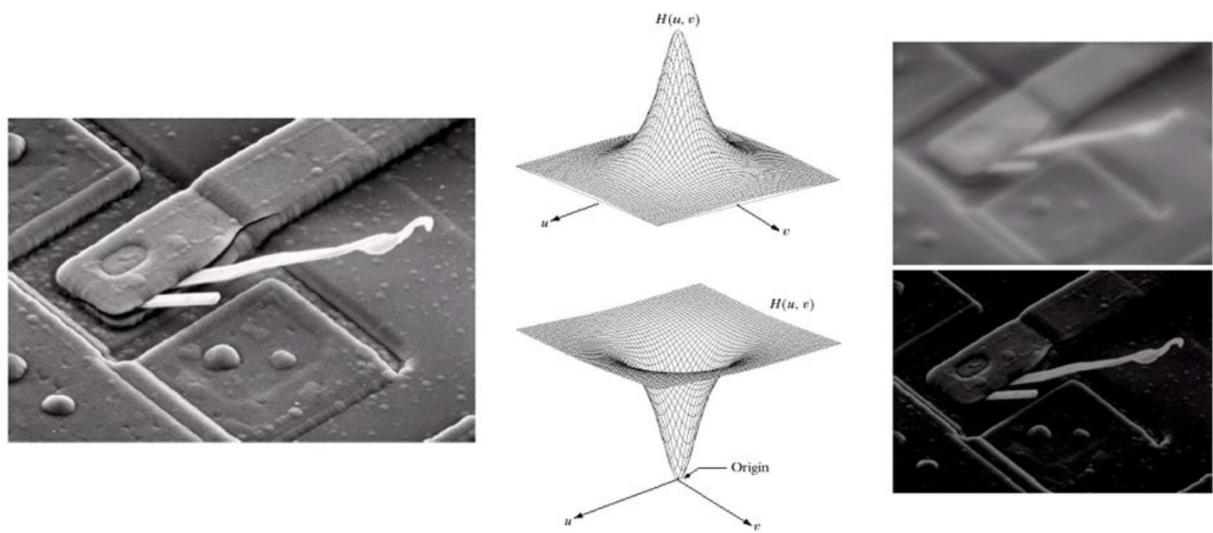


Ilustración 6: Diferencia de filtros de alta y baja frecuencia

Los filtros convolucionales son aún más sencillos y por lo general afectan también al tamaño de la imagen de salida, ya que sólo se centran en partes concretas de la imagen. Su finalidad es extraer un conjunto de características de una misma imagen troceándola en varias subimágenes. Se utilizan muchísimo en tareas de aprendizaje. Estos filtros se explicarán en la sección 4 después de las redes de neuronas.

3.3. Servicios de Computer Vision

Para trabajar en tareas de Computer Vision se requiere disponer de un dataset de ejemplos bastante amplio, lo cuál puede demorar bastante el desarrollo y la posterior evaluación del sistema. Con el objetivo de reducir la curva de aprendizaje y proporcionar herramientas básicas al usuario, grandes compañías como Google, IBM o Microsoft han desarrollado sus propios servicios de Computer Vision.

Google Cloud Vision

Para accede al servicio de Computer Vision de Google debemos acudir al siguiente enlace:

<https://cloud.google.com/vision/>

Ahí podremos observar el diagrama de flujo sobre el que podemos trabajar como usuarios utilizando sus máquinas y datasets. Está releyado también en la siguiente figura

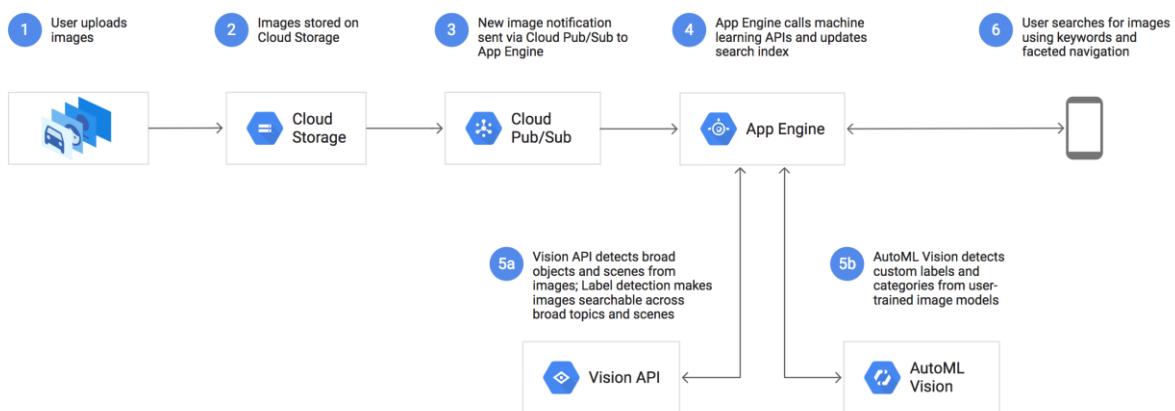


Ilustración 7: Diagrama de flujo de Google Cloud Vision

En el caso de Google, y la mayoría de los servicios de empresas, en primer lugar habría que mandar al servidor la imagen o conjunto de imágenes que queremos utilizar como test. Después se ejecutarían los servicios de visión entre los cuales es habitual encontrar aquel que describe qué hay y qué está pasando en la imagen. A cada palabra asociada a la imagen se le asigna un nivel de confianza entre 0 y 1. Cuanto más alto es el valor, más seguro está el algoritmo de lo que está “observando”. También existen otros servicios como evaluar la peligrosidad de la imagen, el sentimiento de la imagen o las búsquedas relacionadas con ese tipo de imagen en los buscadores.

A continuación, se muestran los cuatro servicios que nos ofrece Google Cloud Vision sólo mediante sus demos [3]. Trabajando a través de su API podemos utilizar aún más servicios [4].

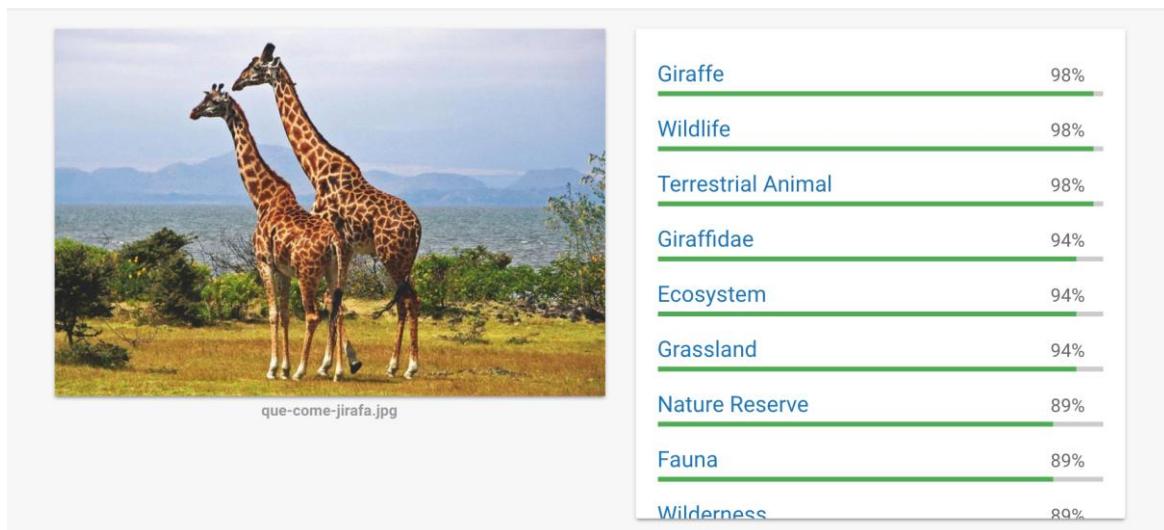


Ilustración 8: Extracción de palabras representativas de la imagen

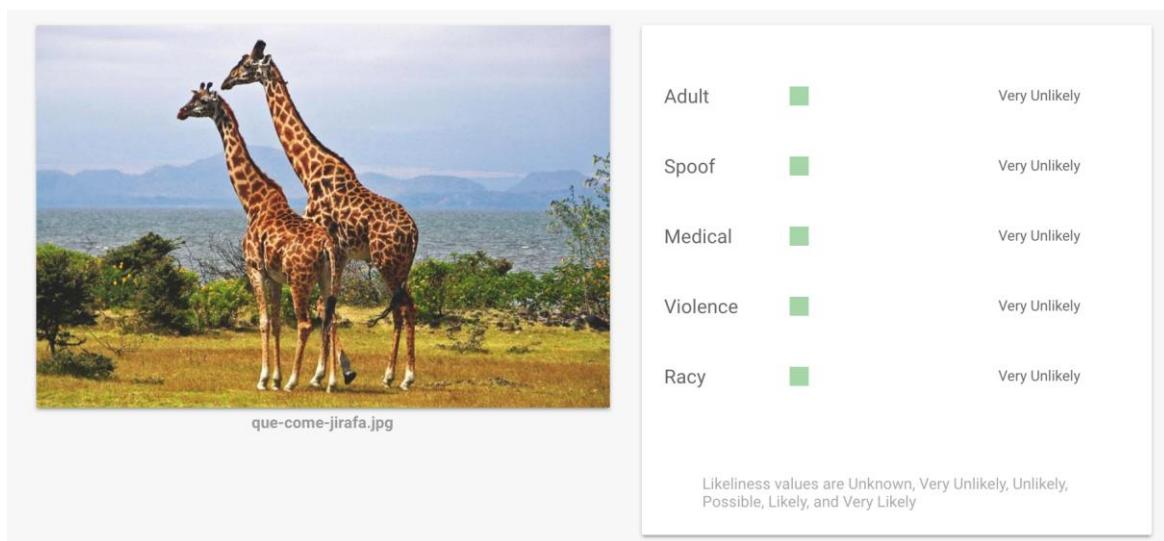


Ilustración 9: Análisis de seguridad de la imagen

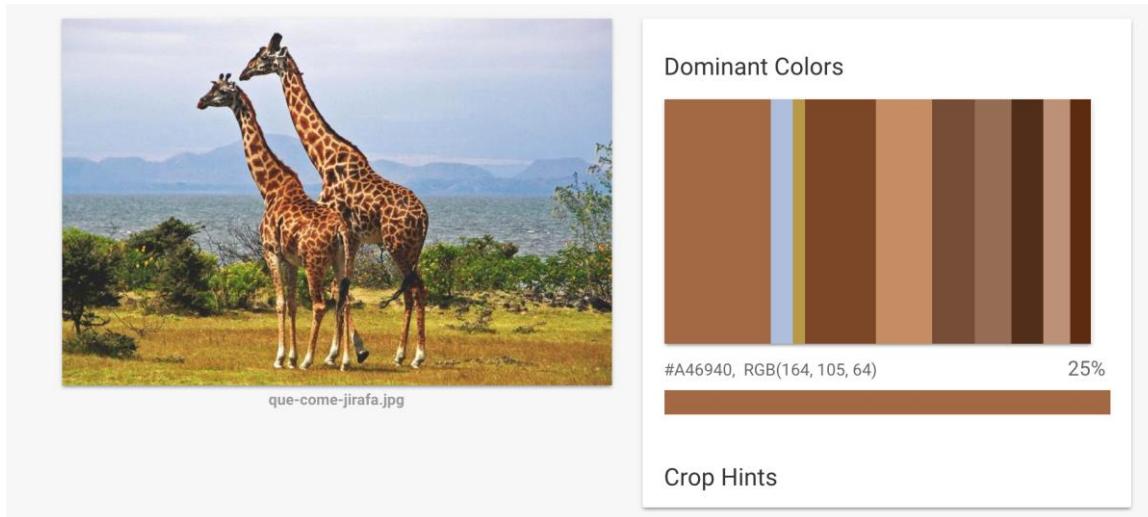


Ilustración 10: Paleta de colores dominantes en la imagen



Ilustración 11: Búsquedas en Internet relacionadas con esta imagen

Otros servicios que pueden consultarse son el modulo de Visual Recognition de IBM Watson [5] y el de Microsoft Azure [6]

4. TAREAS DE APRENDIZAJE EN COMPUTER VISION

Como ya se vio en el tema de Machine Learning, cuando queremos aplicar aprendizaje sobre cualquier tarea dentro del área de la Inteligencia Artificial, necesitamos un algoritmo sobre el que construir nuestro modelo de entrenamiento y de test. En este caso se presentan las redes de neuronas convolucionales como primera solución a realizar tareas de aprendizaje en Computer Vision. Recordamos su definición:

- Redes convolucionales: Las redes de neuronas convolucionales se utilizan principalmente para analizar imágenes y extraer características de estas. Se podría decir que cada píxel de salida es una combinación lineal de los píxeles de entrada. Este tipo de red tiene hasta tres tipos diferentes de capas que se explicarán en la sección 4.7. También varía el tamaño de cada capa de forma decreciente según avanzamos en la red hacia la derecha.
- Redes deconvolucionales: Las redes de neuronas deconvolucionales realizan el proceso opuesto a las convolucionales. De una serie de características e información, amplían cada muestra. En la siguiente figura se entiende mejor el concepto gráficamente.

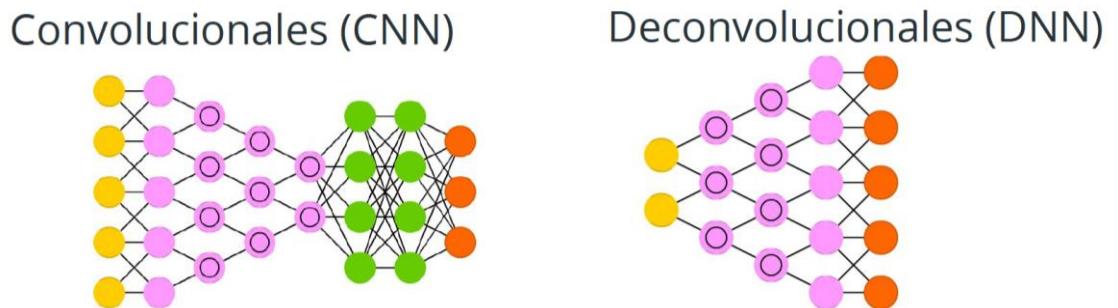
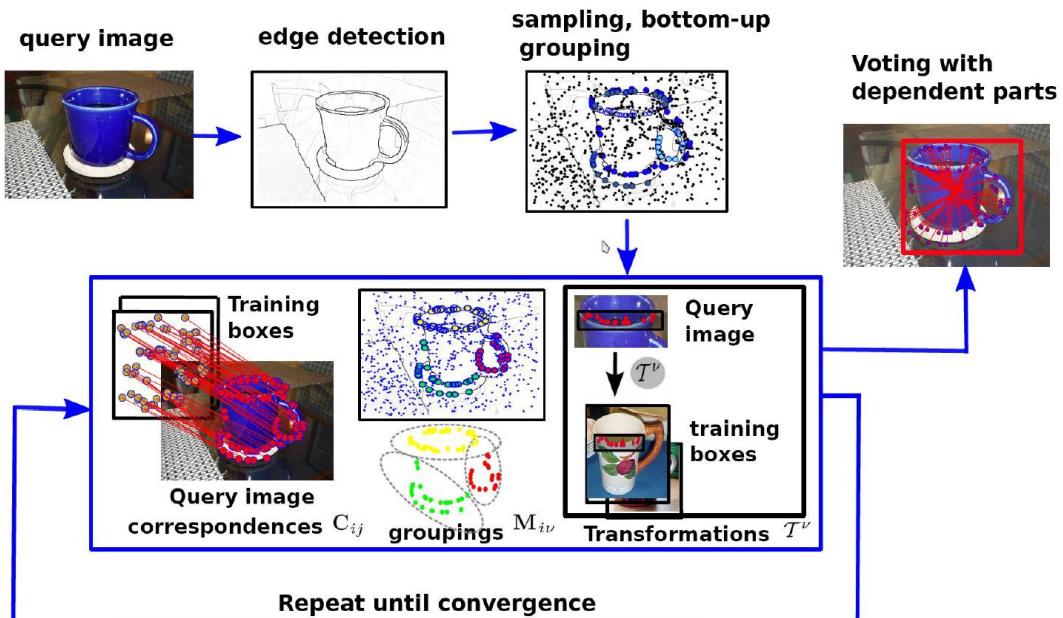


Ilustración 12: Tipos de redes de neuronas

4.1. El proceso de reconocimiento de una imagen

Cuando queremos realizar tareas de aprendizaje utilizando imágenes y Computer Vision, en primer lugar, necesitamos un conjunto de ejemplos con los que podamos comparar la imagen

que vamos a recibir como entrada. En el caso de la figura, dicha imagen sería la llamada *query image* y el conjunto de imágenes sería *training boxes*.



En este caso, la tarea consiste en analizar el objeto que el sistema inteligente tiene delante para posteriormente averiguar cuál es la mejor forma de cogerlo. Una vez se ha capturado la imagen, el sistema aplica el proceso de segmentación y filtrado para reconocer los bordes del objeto y los puntos que pertenecen al mismo y al fondo. Ese conjunto de información generada en torno al objeto se compara con imágenes de tazas que fueron previamente analizadas y que ahora pertenecen al conjunto de entrenamiento. De esta forma se relaciona la información del objeto actual con esa información previa (groupings y query image correspondences) y así se puede validar cómo de exitosa será la salida. Cuanto mayor similitud, mayor probabilidad de éxito. La salida del algoritmo representa la taza con todos esos puntos previamente identificados ahora enlazados entre ellos formando una especie de malla de la parte de la imagen que pertenece al objeto. Este mismo proceso se puede aplicar a cualquier proceso de aprendizaje sobre imágenes. Lo único que cambiará será la tarea que llevar a cabo, la imagen de entrada y el conjunto de entrenamiento.

4.2. Las redes de neuronas convolucionales

Las redes de neuronas convolucionales se utilizan para extraer características e identificar patrones en imágenes. La siguiente figura muestra una representación gráfica de la red típica convolucional así como los distintos tipos de neuronas que intervienen.

En primer lugar, se parte de que se recibe una imagen transformada en matriz como input. Cada píxel está representado por un número de la matriz. La capa de neuronas de entrada recibe esos valores de la matriz directamente, sin ningún tipo de transformación.

La segunda capa de neuronas pertenece a las neuronas kernel. Estas neuronas son las que aplican el primer filtro a los datos de entrada. El kernel no es más que una matriz de reducida dimensión que generalmente está compuesta de 1's y 0's.

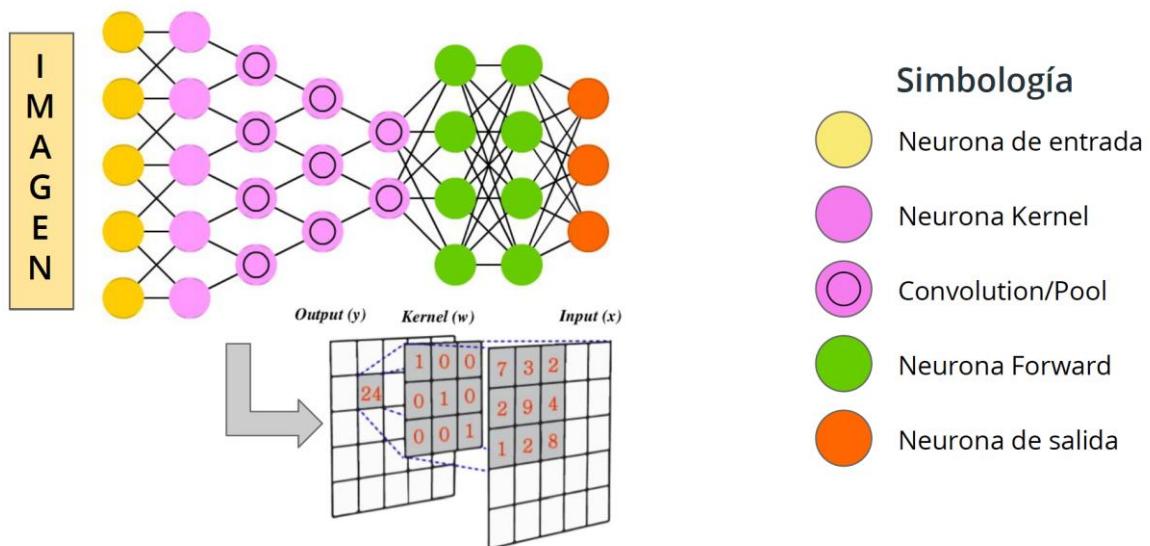


Ilustración 13: Red de neuronas convolucional

Si se analiza el ejemplo proporcionado en la imagen anterior, se puede observar cómo de un subconjunto de 9 valores de la matriz se reduce a un único resultado (24). Esto se consigue aplicando una función kernel en la que sólo la diagonal tiene 1's. Aplicando la multiplicación de matrices Input x Kernel obtenemos ($7 + 9 + 8 = 24$). El objetivo de este paso es reducir la cantidad de información de la imagen de entrada y quedarnos con una muestra de los píxeles. Se pueden utilizar multitud de funciones kernel, esta de la imagen es sólo un ejemplo. Siempre son matrices simples con 0's y 1's para reducir rápidamente la cantidad de información. El proceso se repite para cada zona de la imagen input desplazando la matriz kernel hacia posiciones no analizadas. En la siguiente figura se muestra una ampliación de cómo estarían

superpuestas la imagen de entrada (matriz azul) y la que obtenemos como salida de la función kernel (matriz turquesa). La submatriz azul oscura representa la parte afectada por la función kernel (la matriz de 0's y 1's). La submatriz turquesa oscuro representa el número resultante de aplicar la función kernel.

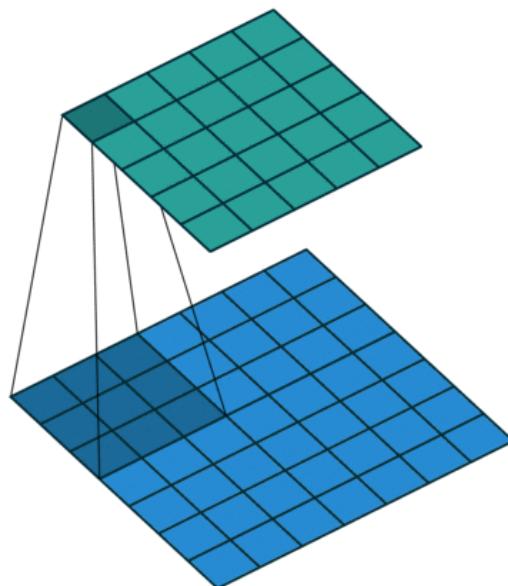


Ilustración 14: Aplicación de la función kernel

Una vez se ha realizado el muestreo, comienza el análisis de características de la imagen muestreada. Para ello, llamaremos a esta imagen o conjunto de imágenes si son varias, mapa de características (feature map) e iremos extrayendo partes del mismo de forma encadenada como se muestra en la siguiente figura:

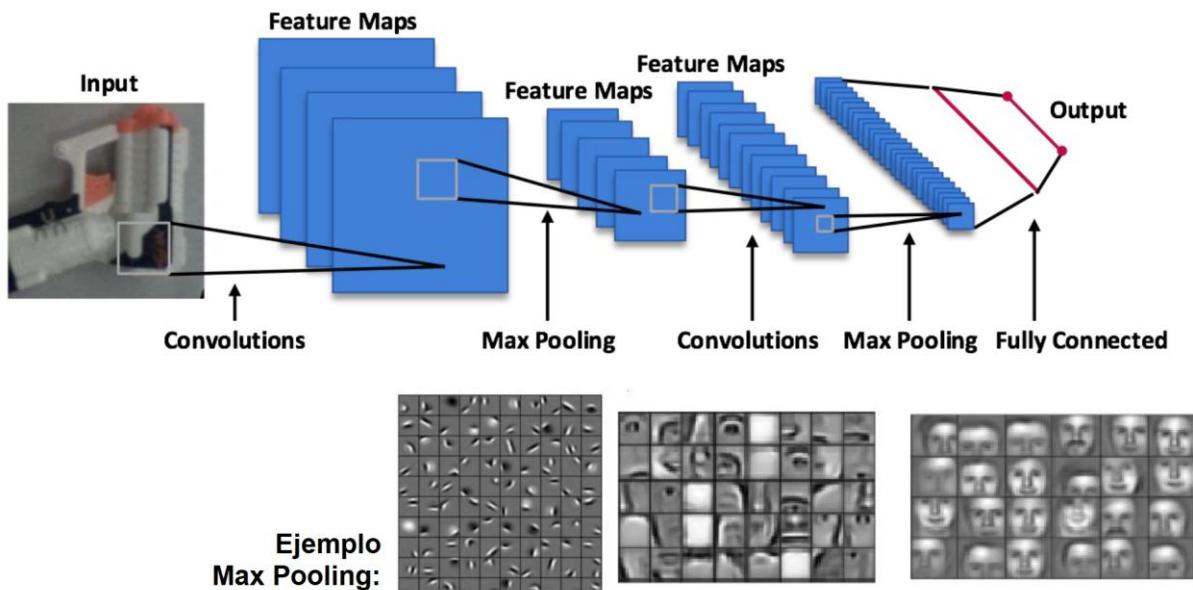
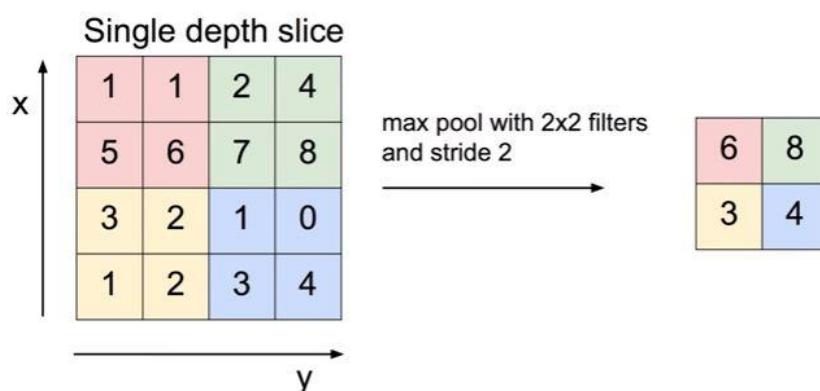


Ilustración 15: Red de neuronas convolucionales (max pooling)

La extracción de características se lleva a cabo mediante el filtro *max pooling*. Dicho filtro nos permite hacer “zoom in” y “zoom out” a la propia imagen muestreada o mapa de características. Si se observa el ejemplo, se apreciará cómo se pasa de obtener un detalle muy pequeño de la imagen (líneas de alto contraste) a ojos y después rostros.

El filtro max-pooling ofrecerá un resultado u otro dependiendo de su tamaño (ancho x alto) y desplazamiento sobre la imagen muestreada (stride). El comportamiento es similar al de la función kernel, aunque en este caso el filtro opta por coger el mayor valor de cada región. En el siguiente ejemplo, de un filtro max pool de 2x2 y desplazamiento 2 obtendríamos los valores 6, 8, 3 y 4; ya que son los valores máximos de cada submatriz.



Las redes convolucionales se construyen mediante combinaciones de capas convolucionales (funciones kernel), de max pooling y de una función de activación llamada ReLu (rectified linear units). Dicha función descarta cualquier valor negativo que haya podido propagar la capa de convolución o la de max pooling sustituyendo éstos por cero.

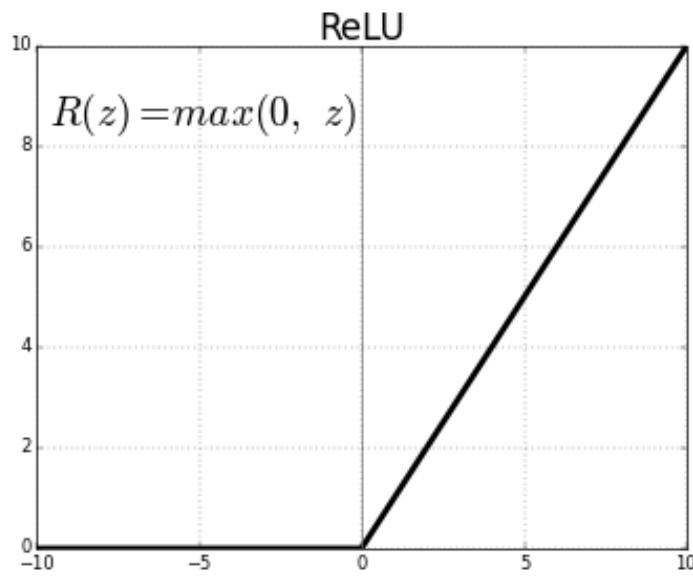


Ilustración 16: Función ReLU

Si se observan de nuevo las imágenes 14 y 16, se aprecia un último paso en el que una capa de tipo feedforward se encarga de propagar los últimos valores obtenidos a través de max pooling a la salida. Generalmente, la salida de la red convolucional es una palabra o conjunto de palabras que identifican lo que contiene la imagen de entrada.

Hay que tener en cuenta que la última capa de max pooling tiene unas dimensiones diferentes a la de feed forward (o fully connected), por lo que habrá que realizar una redimensión de los datos.

Por último, se muestra a continuación en la siguiente figura todo este proceso recogido de forma abstracta de identificación de un coche. En la parte inferior se observan las diferentes características que va extrayendo la parte convolucional de la red. La salida devuelve cinco valores, por lo que la última capa tendrá cinco neuronas. La neurona con mayor peso equivale a la de la categoría que se ha identificado como más probable (car).

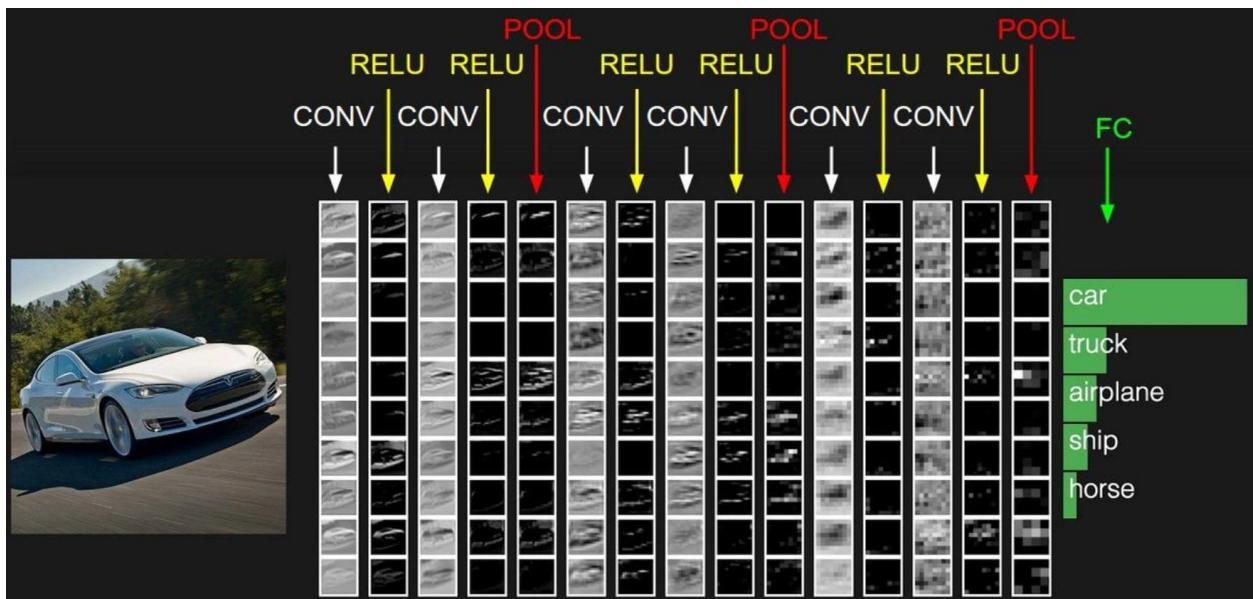


Ilustración 17: Ejemplo de análisis de imagen utilizando una red convolucional

Las redes de neuronas se pueden trabajar con diferentes librerías y lenguajes de programación. La mayoría de las personas que se dedican a este campo utilizan Python y R para programarlas. Las librerías más populares son TensorFlow [7] y IBM Watson [5] (la parte cognitiva).

5. EL RECONOCIMIENTO DEL HABLA

Dentro de la Inteligencia Artificial (IA) existe un campo que se preocupa por la interacción humano-máquina a través de « los procesos del lenguaje natural », o lo que es lo mismo, el reconocimiento del habla. Por este motivo, el área de la lingüística aplicada a la IA genera este nuevo campo de conocimiento al que llamaremos Natural Language Processing (NLP).

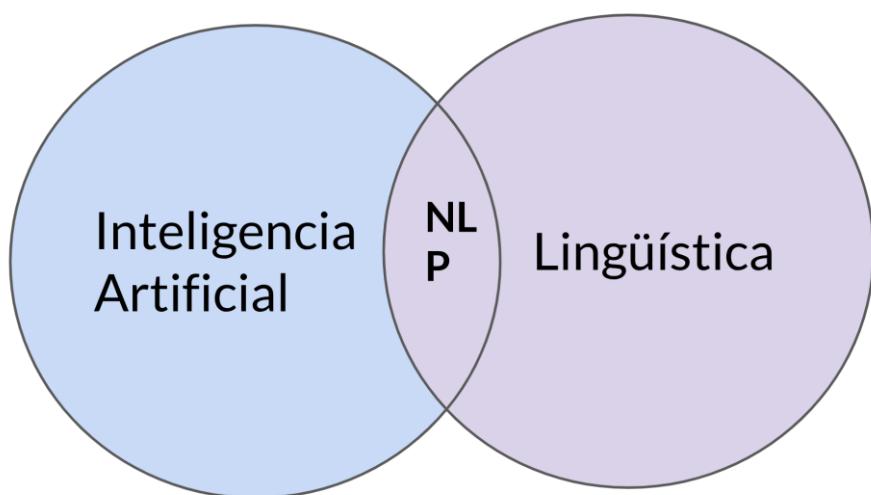


Ilustración 19: NLP representa la intersección entre la IA y la lingüística

Se define entonces el Natural Language Processing como el área de la Inteligencia Artificial que se encarga de estudiar la interacción humano-máquina a través del lenguaje (hablado o transscrito). En concreto, los investigadores se centran en cómo elaborar programas o módulos que sean capaces de analizar y entender el lenguaje y sus expresiones.

Las aplicaciones de NLP dentro de los sistemas inteligentes son múltiples. En concreto, dentro del campo de la robótica, las principales tareas donde se trabaja con técnicas de procesamiento del lenguaje natural son:

- Traducción de textos
- Transcripción de textos
- Análisis sintáctico-semántico
- Recuperación de información

- Análisis del sentimiento
- Elaboración de pregunta-respuesta
- Expresividad y entonación

5.1. Los principales elementos del Natural Language Processing

Al igual que sucedía con las tareas de Computer Vision, para trabajar en Natural Language Processing necesitamos un dataset de ejemplos, pues de alguna forma necesitamos reconocer el habla y después procesarlo. En este campo, a los datasets se les llama corpus. Dicho nombre se hereda del propio campo de la lingüística.

Un *corpus* es un conjunto de textos que comparten idioma y temática (si son más específicos). Se almacenan electrónicamente y se procesan a modo de análisis, validación y verificación de reglas lingüísticas.

Existen numerosos tipos de corpus: textos agrupados por idioma, por temática específica (legal, educativa, medicina...), por ideología, por sentimiento... según la tarea que queramos resolver, el corpus será más complicado o más sencillo de confeccionar.

Debido a la complejidad de obtener variedad de textos, muchos desarrolladores optan por utilizar técnicas de scrapping sobre portales especializados en una temática. El scrapping consiste en desarrollar un programa que te permita recopilar toda la información de una web, generalmente almacenada con la misma estructura y con poca variabilidad en cada una de sus URL. Por ejemplo, si se quiere hacer un módulo especializado en pintura, se podría acceder a la biblioteca web de El Prado o El Thyssen y obtener las fichas de cada uno de sus cuadros.

Los lenguajes de programación más populares para resolver tareas de NLP son R y Python.

6. EL PROCESO DE RECONOCIMIENTO DEL LENGUAJE

Para conseguir la interacción entre humanos y máquinas a través del habla, primero hay que establecer los pasos que le permitan a la máquina reconocer el lenguaje y procesarlo. Se encuentran recogidos en la siguiente figura.

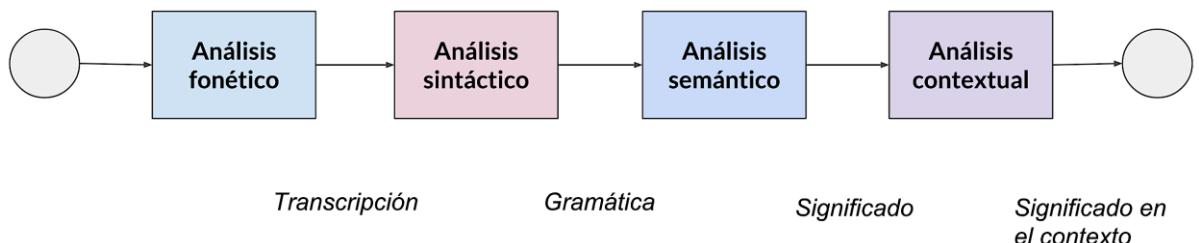


Ilustración 180: El proceso de reconocimiento del habla

A continuación, se describe brevemente cada uno de los pasos:

- Análisis fonético: si la entrada se recibe en forma de audio es necesario contar con esta fase en la cual el sistema analizará fonéticamente dicha pieza. El objetivo es transcribir el audio para que el resto las fases ya partan de un texto a analizar. Recuerda que siempre operamos de la misma forma: cualquier entrada que no sea puro texto o números siempre se transforma previamente a alguna de estas dos categorías.
- Análisis sintáctico: una vez se recibe la cadena en forma de texto, el primer paso será analizar sintácticamente cada uno de sus elementos. Esto es, evaluar el componente gramatical de cada elemento y las estructuras gramaticales que se forman en el texto.
- Análisis semántico: este segundo análisis complementa al sintáctico, ya que el objetivo es evaluar cada tipo de elemento del texto (sustantivo, adjetivo, adverbio...) así como su género, el número, si es formal o informal etc.
- Análisis contextual: finalmente, el análisis contextual nos permite analizar el texto en global y entender el propio contexto en el que se está dando lugar. Este paso es el más complejo para la máquina debido a que generalmente se espera una respuesta por parte de la misma que sea acorde al contexto.

Antes de continuar con las diferentes técnicas que se utilizan para identificar el idioma u otros muchos componentes clave en un texto, se muestra un ejemplo de análisis sintáctico y semántico obtenido a través de la herramienta AutoML de Google.

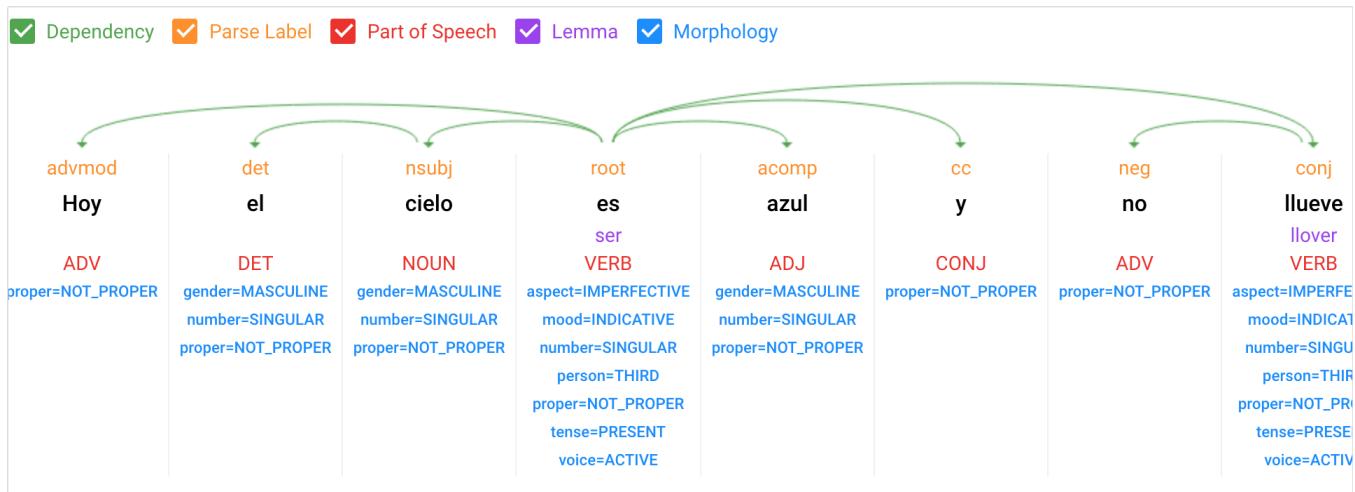


Ilustración 21: Ejemplo de análisis sintáctico-semántico

6.1. Analizando el sonido del habla

A la hora de convertir el audio en texto, los sistemas de procesamiento deben tener en cuenta los siguientes aspectos:

Frecuencia del habla o *pitch*: existen unos intervalos medios de frecuencia para las voces de hombres y mujeres que hay que tener en cuenta para reconocer una misma frase independientemente de quién sea el orador.

Hombres: 85 a 180 Hz

Mujeres: 165 a 255 Hz

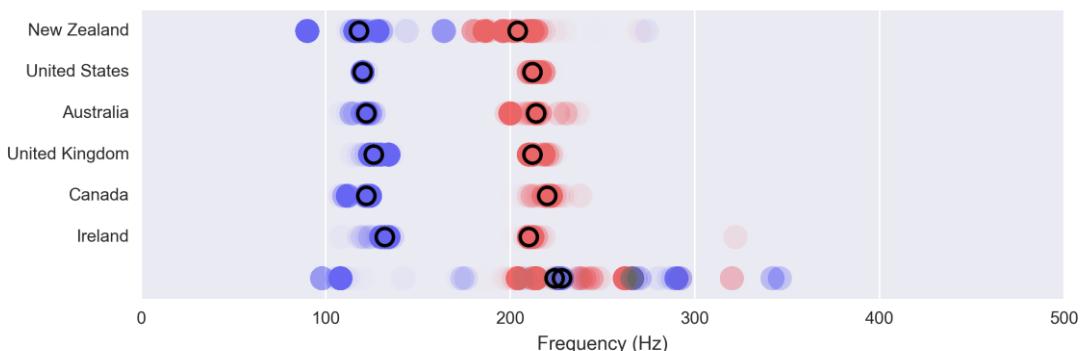


Ilustración 22: Frecuencias de voz en países de habla inglesa

Además, dependiendo del idioma, existen algunas variaciones en la frecuencia, aunque son menores. En las dos figuras se representan los países de habla inglesa y de habla hispana con

sus correspondientes rangos de frecuencia para mujer (rojo) y hombre (azul). Se puede leer más sobre este análisis en [8].

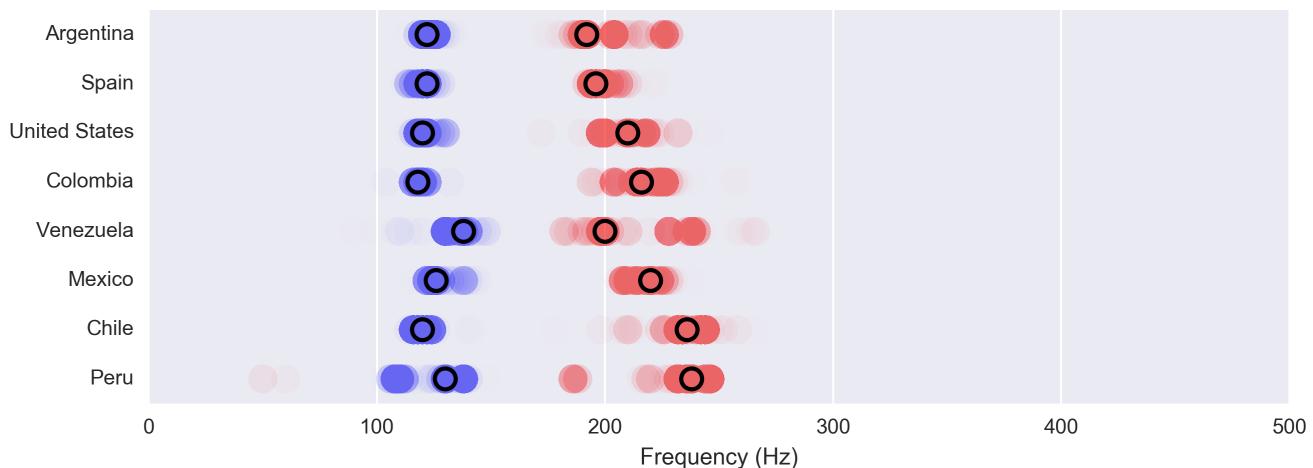


Ilustración 23: Frecuencias de voz en países de habla hispana

Por otro lado, en esta fase de transcripción del habla también hay que tener en cuenta la entonación (tone) y las pausas (stress).

6.2. Text-mining

Un texto se considera información no estructurada desde el punto de vista de un sistema de inteligencia artificial o aprendizaje automático. Por este motivo, las técnicas de text-mining se encargan de estructurar y analizar cualquier texto recibido como entrada con el fin de que sea fácilmente procesable por los algoritmos de aprendizaje automático. A continuación, se describen cuatro técnicas populares en text-mining:

Tokenization: consiste en dividir una secuencia de texto en frases o palabras llamadas tokens. Al mismo tiempo, se suelen eliminar también signos de puntuación. Con la lista de tokens ya se dispondría de una primera versión estructurada del texto.

Filtering: esta técnica se aplica sobre documentos para eliminar algunas palabras. En concreto, es muy común aplicar el filtro stop-words. El conjunto de stop-words de un texto son aquellas palabras que aparecen frecuentemente y no tienen demasiada carga semántica, como por ejemplo las preposiciones, las conjunciones etc. A su vez, también se pueden eliminar palabras que aparezcan una sola o un par de veces y que a pesar de tener carga semántica no aporte demasiado su contribución al contexto textual.

Lemmatization: consiste en realizar un análisis morfológico de cada palabra del texto. Si hay palabras repetidas, sólo se analizan una vez. Se trata también de agrupar las palabras que tienen la misma raíz gramatical con el fin de convertir las diferentes formas verbales a infinitivos y los sustantivos a una misma forma gramatical. Como esta técnica suele ser bastante tediosa de aplicar, especialmente si se debe aplicar a un conjunto grande de documentos, se prefieren las técnicas de stemming.

Stemming: esta técnica consiste en obtener la raíz (stem) de todas las palabras derivadas de un texto. Como consecuencia, estos métodos y la forma de aplicarlos dependerán del idioma en el que esté escrito el texto. En [9] se puede encontrar uno de los stemmers más utilizados en textos de lengua inglesa.

Finalmente, una vez hemos aplicado tokenization + filtering + (lemmatization o stemming) dispondremos también de las palabras más repetidas y de mayor carga semántica del texto o de los textos. De esta forma, podemos inferir el contexto del texto.

6.3. Métricas

Dado un conjunto de documentos $D = \{d_1, d_2, \dots, d_D\}$, y disponiendo de otro conjunto vocabulario que represente las palabras de dicho conjunto (eliminando repetidas) $V = \{w_1, w_2, \dots, w_v\}$. La frecuencia de una palabra $w \in V$ perteneciente a un documento $d \in D$ se representa mediante $f_d(w)$ y el número de documentos que contienen dicha palabra se representa mediante $f_D(w)$. En text-mining es muy común que cada palabra se represente con una variable (peso) al que se le asigna un valor numérico que indica la importancia de dicha palabra en el documento. Dicho peso puede computarse de diversas formas. A continuación presentamos dos de ellas:

- 1) Método booleano: en este caso se asigna un peso $\omega_{ij} > 0$ a cada palabra perteneciente a un documento $w_i \in d_j$. Si la palabra no aparece en el documento d_j , entonces $\omega_{ij} = 0$.
- 2) Term frequency-inverse document frequency (TF-IDF): podría traducirse como la frecuencia de ocurrencia del término en la colección de documentos, a lo que

llamaremos $q(w)$. TF-IDF es la métrica más popular de los trabajos relacionados con text-mining. Se computa de la siguiente forma:

$$q(w) = f_d * \log \frac{|D|}{f_D(w)}$$

6.4. Sentimiento del texto

El análisis del sentimiento de un texto o conjunto de textos es el proceso utilizado para averiguar la emoción que hay detrás del mismo. En concreto se analizarán conjuntos de palabras o frases y se realizará un análisis de la evolución del sentimiento.

Este tipo de análisis se utiliza mucho en redes sociales asociado a marcas, con el fin de saber cómo los perciben los usuarios, o a la hora de estrenar música, series, películas, para saber cómo reacciona o reaccionará la audiencia. También es muy popular su uso en páginas de reseñas de moda, viajes, grandes almacenes etc.

Como sucedía con técnicas anteriores, para realizar el análisis del sentimiento necesitamos un dataset. En este caso, los datasets se llaman lexicón, ya que contienen las palabras y palabras derivadas de un idioma asociadas a un sentimiento que, por lo general, siempre se mantiene independientemente del contexto.

Un ejemplo de extracto de un lexicón en inglés es el siguiente que podemos ver aquí:

```
##   word      sentiment lexicon score
##   <chr>    <chr>     <chr>   <int>
## 1 abacus    trust      nrc      NA
## 2 abandon   fear       nrc      NA
## 3 abandon   negative   nrc      NA
## 4 abandon   sadness    nrc      NA
## 5 abandoned anger     nrc      NA
## 6 abandoned fear      nrc      NA
## 7 abandoned negative   nrc      NA
## 8 abandoned sadness    nrc      NA
## 9 abandonment anger    nrc      NA
## 10 abandonment fear     nrc      NA
```

De nuevo, ésta es una técnica que depende mucho del idioma para poder explotarse. La mayoría de diccionarios libres y gratuitos se han desarrollado para el inglés. Aquí se pueden consultar tres:

- AFINN [11]
- Bing [12]
- Nrc [13]

Finalmente, para profundizar más en este tema a nivel de código se puede seguir el tutorial presentado en [10]. Utiliza el lenguaje R y comienza desde los temas de text-mining más sencillos hasta los más complejos, como sería un sistema final de análisis de textos. También se pueden consultar los diferentes datasets para NLP recopilados en [14].

6.5. Diferencias entre text-mining y natural language processing

A pesar de haber explicado las técnicas de text-mining dentro de este tema de Natural Language Processing, hay que remarcar que dichas técnicas están orientadas al análisis de documentos y no directamente al análisis del habla. Si bien es cierto, las técnicas de text-mining nos pueden ayudar a evaluar el sentimiento o el contexto de la frase que estemos analizando en NLP. Además, también se pueden utilizar para detectar y resolver las ambigüedades del lenguaje como:

- **Ambigüedad léxica:** palabras con múltiples significados.
- **Ambigüedad sintáctica:** frase que se puede analizar sintácticamente de distintas formas.
- **Ambigüedad semántica:** frase con múltiples significados
- **Ambigüedad anafórica:** frase o palabra que se menciona previamente pero que cobra un significado diferente.

7. EL PROCESAMIENTO DEL LENGUAJE NATURAL

Para trabajar en sistemas de aprendizaje de Natural Language Processing, se utilizan las redes de neuronas recurrentes. Estos modelos matemáticos nos permiten por ejemplo elaborar teclados con lenguaje predictivo, como los que se utilizan en los smartphones, o elaborar respuestas mediante un chatbot que resuelva dudas a un potencial cliente.

La mayoría de sistemas de aprendizaje NLP siguen la arquitectura definida en la siguiente figura.

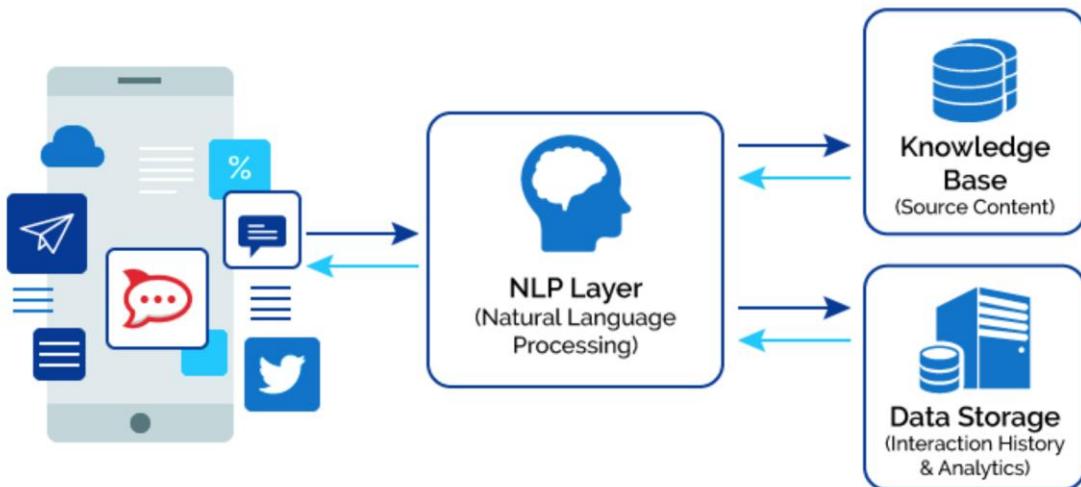


Ilustración 25: Arquitectura de los sistemas NLP

En primer lugar, se necesita una base de conocimiento. Previamente los hemos llamado datasets, conjuntos de entrada, corpus, etc. Dependiendo del problema a resolver la base de conocimiento será más grande o más pequeña y más o menos específica. También necesitaremos una base de datos para almacenar todo el histórico de datos estructurados, organizados y analizados. Aquí también se encontrarán los datos utilizados para entrenar el sistema.

Tanto la base de conocimiento como la base de datos están ligados a la capa de NLP, pues el sistema la utiliza para verificar la información mediante las técnicas explicadas en secciones anteriores y el proceso explicado en la sección 3.

La capa de NLP es la encargada de comunicarse con el elemento con el que también se comunica el usuario. Ésto puede ser un smartphone, un robot, un chatbot, un altavoz como Alexa o Google Home etc.

7.1. Servicios de NLP

Para trabajar en tareas de Natural Language Processing se requiere disponer de un corpus y lexicon bastante amplio, lo cuál puede demorar bastante el desarrollo y la posterior evaluación del sistema. Con el objetivo de reducir la curva de aprendizaje y proporcionar herramientas

básicas al usuario, grandes compañías como Google, IBM o Microsoft han desarrollado sus propios servicios de Natural Language Processing.

Google Cloud Natural Language

Para acceder al servicio de Natural Language de Google debemos acudir al siguiente enlace:

<https://cloud.google.com/natural-language/>

Ahí podremos encontrar servicios de análisis sintático y semántico de textos, análisis del sentimiento, análisis morfológico, texto-voz y voz-texto, traducción simultánea de textos, etc.

La mayoría de las tareas se encuentran en la página web mencionada anteriormente, aunque hay otras que están divididas entre:

<https://cloud.google.com/speech/>

<https://cloud.google.com/translate/#how-automl-translationbeta-works>

Como demo, Google muestra la capacidad de análisis sintáctico-semántico y de sentimiento, en diferentes idiomas. Mostramos a continuación dos ejemplos en castellano:

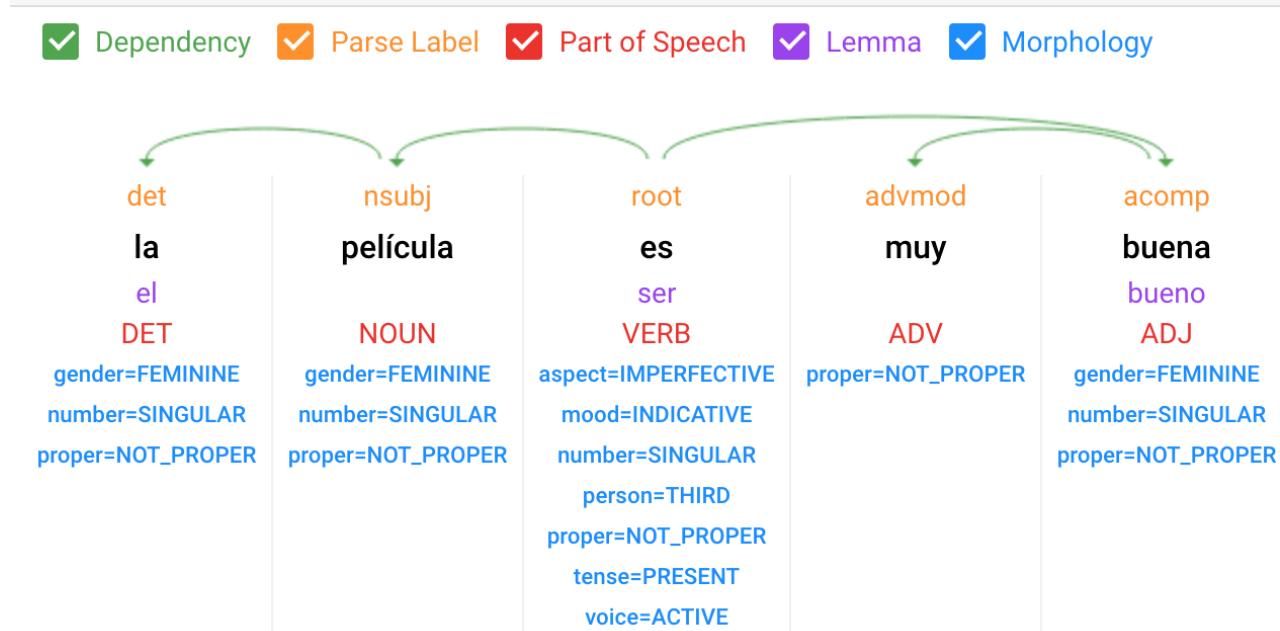


Ilustración 26: Ejemplo de análisis sintáctico-semántico con Google Natural Language



Ilustración 27: Análisis del sentimiento con Google Natural Language

En el caso concreto de NLP, la empresa IBM tiene disponibles bastantes servicios como traducción, voz-texto y viceversa, análisis de la entonación, estilos de personalidad etc. Se pueden consultar todos en [15].

7.2. Redes de neuronas recurrentes

Las redes de neuronas, que se introdujeron en el tema anterior, se utilizan en múltiples campos. Uno de ellos es el de Natural Language Processing. Recordemos que la estructura de la red de neuronas pasa por una capa de entrada, una serie de capas ocultas y una capa de salida.

En concreto, vamos a hablar de dos tipos de redes:

- Redes Elman (simple feedback): Las redes de neuronas Elman son denominadas redes recurrentes simples ya que incluyen retroalimentación entre las capas contiguas. Es decir, en este tipo de red las neuronas poseen una memoria de los eventos inmediatamente anteriores. Esta información es utilizada en el proceso de actualización de los pesos durante el proceso de aprendizaje. En la siguiente figura se puede apreciar la estructura de una red Elman.

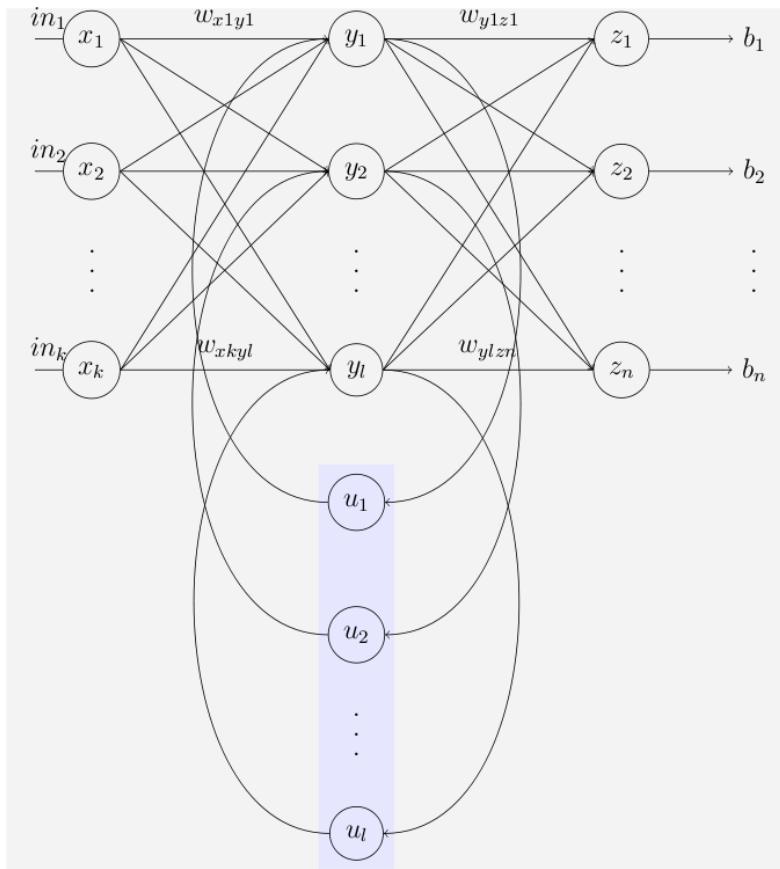


Ilustración 28: Red Elman

De forma más esquemática podríamos representar a la neurona recurrente como en esta otra figura. La neurona equivale a la caja verde A. La entrada a la neurona es el círculo X_t y la salida el círculo h_t . En cada iteración ($0 \dots t$), la neurona irá reteniendo el conocimiento que ella misma ha inferido y lo sumará al que ya contiene.

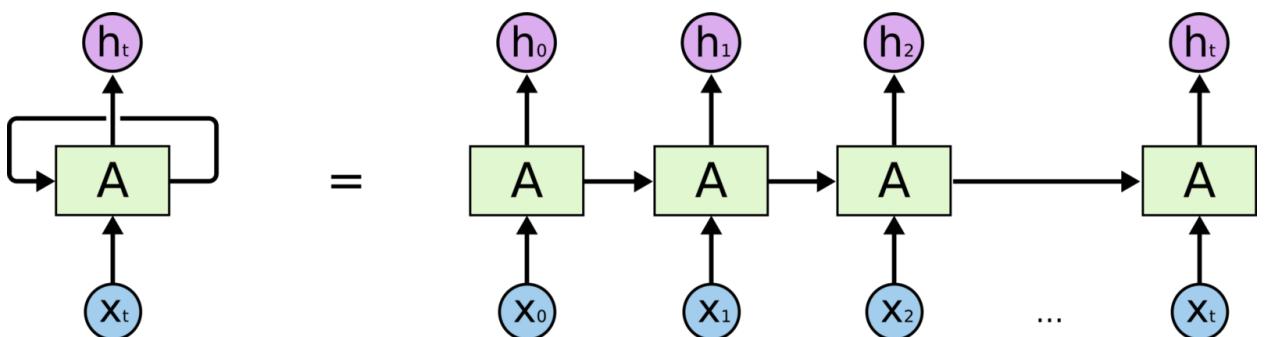


Ilustración 29: Neurona recurrente

Como se ha mencionado, debido a la conexión cíclica de estas redes, son muy buenas retenedoras de conocimiento en el corto plazo. Esto significa por ejemplo que retienen muy bien relaciones entre sustantivos y adjetivos. Retomando la frase del ejemplo de análisis

sintáctico “Hoy el cielo es azul”, si ésta se guardase en una red de neuronas recurrente en la entrada se recibiese en castellano, sería muy fácil que la red infiriese que después de cielo + verbo el adjetivo más propenso a aparecer es azul.

Las redes clásicas feed-forward, representadas como en la parte derecha de la siguiente imagen, no contienen este ciclo así que el conocimiento que retienen se obtiene en el largo plazo. Esto también se convierte en una carencia de las propias redes recurrentes.

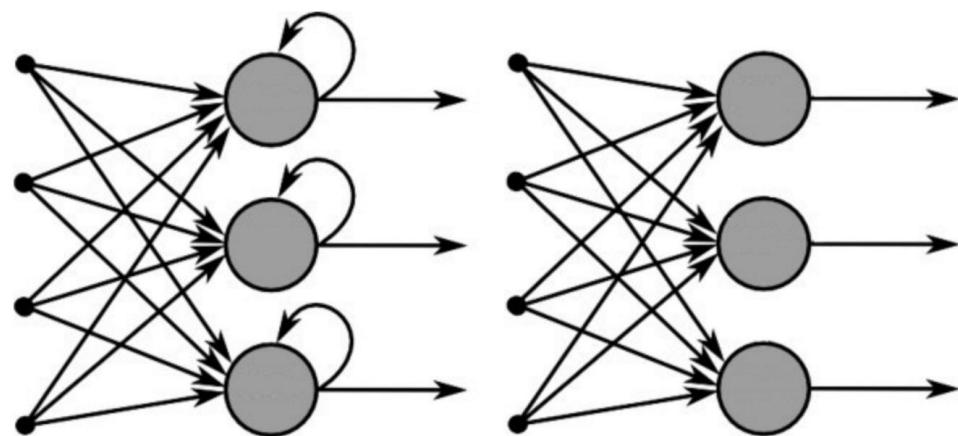


Ilustración 30: Recurrente vs Feed-forward

Para pulir dicha carencia y ampliar las tareas que puedan resolver las redes de neuronas recurrentes, surgen otro tipo de redes más completas y complejas: las redes full feedback.

- Redes recurrentes (full feedback): Las redes de neuronas recurrentes completamente conectadas, a diferencia de las redes de neuronas Elman, tienen conexiones de retroalimentación a todos los elementos que forman la red. Cada neurona de la red está conectada a todos los elementos que la rodean desde un punto de vista espacial. Es decir, una neurona está conectada a las neuronas de las capas posteriores y anteriores y a si misma.

La particularidad de las redes full feedback es que además la capa de salida está conectada a la entrada de cada neurona recurrente. En cualquier caso, existe otro modelo basado en las full feedback mucho más popularizado en los últimos años: las redes Long-Short Term Memory.

Antes de explicar las redes Long-Short Term Memory (LSTM) se presentan tres ejemplos para que se entienda bien su principal funcionalidad: cubrir la memoria en el corto y en el medio plazo.

Una de las principales funciones de las redes recurrentes en general es la retención de patrones/conocimiento en el corto plazo. Como se presentó anteriormente, esto influye a nivel de palabra, a nivel de letras o a nivel de estructuras gramaticales siempre que se encuentren relativamente cerca unas de otras.

El primer caso podemos verlo en la figura de la derecha: el teclado predictivo a nivel de letras. Una vez se ha escrito la raíz de la palabra es muy sencillo para el sistema inteligente proponerte la alternativa que mejor encaja gramaticalmente hablando (-ion) y la segunda más utilizada, que sería en forma de verbo terminado en -ing. La alternativa con mayor probabilidad de éxito es la que siempre se sugiere en el centro de la pantalla.

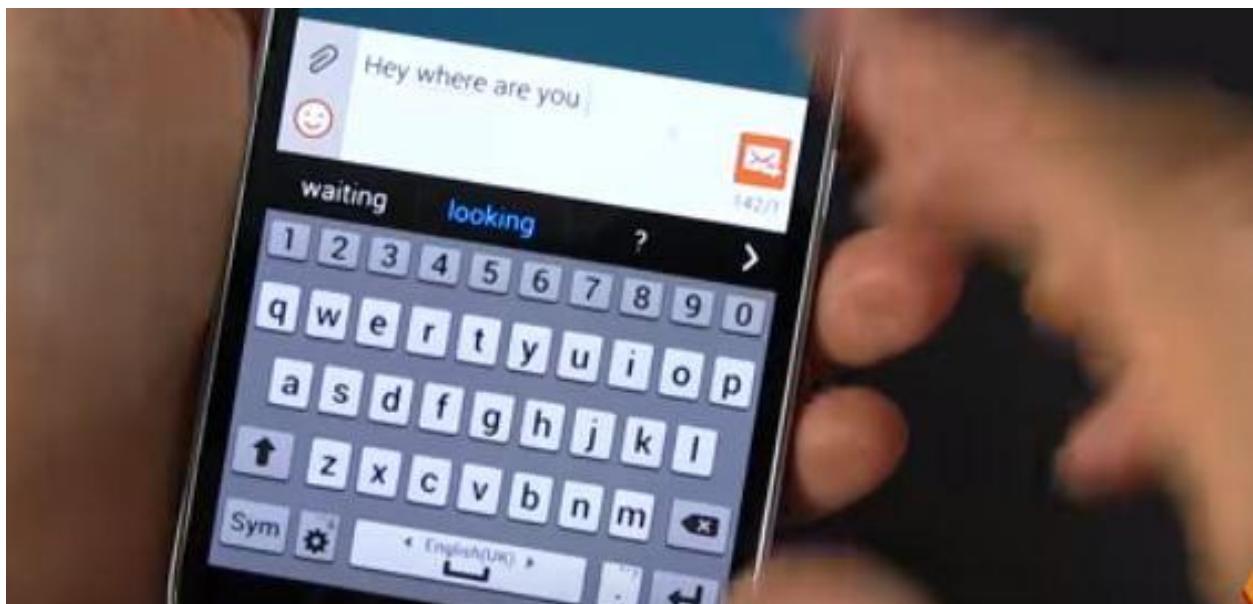


Por otro lado, el segundo ejemplo lo encontramos también en el teclado predictivo cuando hemos escrito unas palabras y aún nos queda alguna para completar la frase. En la siguiente figura aparece la frase "Hey where are you..." ("Oye, ¿a dónde estás...."). El sistema inteligente sugiere "looking" como mayor probabilidad de éxito y "waiting" y "?" como segundas.

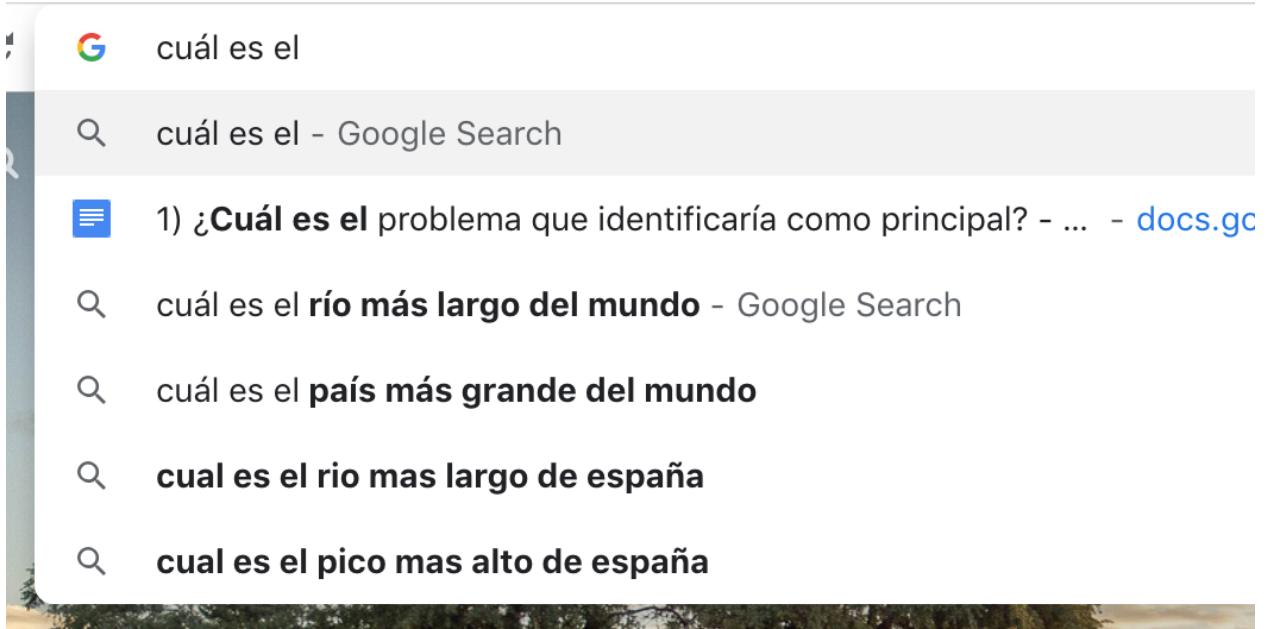
- Hey where are you looking..
- Hey where are you waiting... (Oye, ¿dónde esperas...?)
- Hey where are you? (¿Oye, dónde estás?)

Dado que el verbo (where) va antes que el sujeto (tú) el algoritmo deduce que probablemente se trate de una pregunta; por eso sugiere "?" entre los candidatos, para finalizar la estructura.

Los casos de looking y waiting son más particulares del propio usuario. Generalmente, según como escribamos y qué palabras utilicemos habitualmente, el algoritmo sugerirá unas palabras u otras. Si bien es cierto que en este caso looking y waiting tienen relación con el contexto de la pregunta, que una esté encima de otra en el ranking puede venir dado por la cantidad de veces que se haya utilizado un respecto a la otra.



Finalmente, el último caso se puede apreciar fácilmente en el explorador de Internet; ocurre a nivel de frase. En este caso, si escribo en mi navegador “Cuál es el....”, el primer resultado que obtengo es un documento que he creado yo en mi cuenta cuyo nombre es similar según la consulta realizada. La propia información que tenga almacenada en la cuenta siempre se mostrará en los primeros puestos, dado que equivaldría a facilitar un trabajo de recuperación y acceso a la información. Después obtengo las consultas más populares del buscador referidas a mi media pregunta. Si te fijas con atención verás que las preguntas sugeridas están relacionadas con “el mundo” (preguntas más populares) y España (lugar en el que resido), por lo que se puede concluir que uno de los elementos que relacionan las propias redes es la ubicación desde donde estén haciendo la pregunta.



Estas tres tareas de predicción contienen relaciones inmediatas (corto plazo) y otras a medio plazo, como puede ser la última tarea en la que se tiene en cuenta el contexto del usuario. Cuando trabajamos sobre tareas de predicción o traducción en NLP, utilizamos las redes LSTM ya que equilibran mejor la retención de información en el corto y medio plazo.

7.3. Redes LSTM

Las redes Long-Short Term Memory (LSTM) son redes de neuronas recurrentes capaces de preservar el error que ese propaga a través de las distintas capas e iteraciones durante bastante tiempo (más de 100 iteraciones). Como consecuencia, estas redes son capaces de establecer relaciones de causalidad (causa-efecto) entre los elementos analizados. Fueron inventadas por [16] en el año 2000.

Las neuronas clásicas propagan toda la información de izquierda a derecha y generalmente tienen un peso asociado que les indica su propia relevancia en la red. Por el contrario, las neuronas de las redes LSTM contienen una estructura más compleja, en la que la información de salida puede volver a la entrada (como ya ocurría con las redes recurrentes), pero además puede ser modificada u olvidada. Esto se realiza mediante un conjunto de funciones matemáticas, operaciones lineales y elementos lógicos llamados puertas. Mediante las puertas,

se controla la cantidad de información que entra, sale o se olvida. Las puertas utilizan funciones sigmoidales para propagar la información. Existen diferentes formas de representar e implementar una neurona recurrente. En la figura 12 se muestra una de ellas y se muestra también la propia comparación con la neurona recurrente clásica.

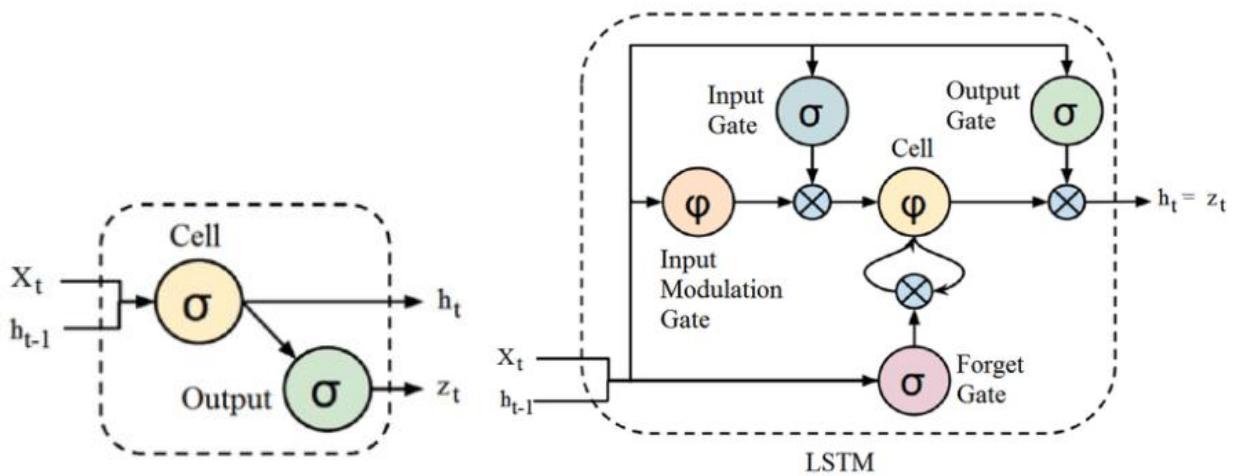


Ilustración 31: Neurona recurrente vs Neurona LSTM

Como se puede observar, la neurona LSTM tiene una estructura mucho más compleja que la recurrente clásica. Una de las principales diferencias está en la salida. Llamamos h_t a la salida que vuelve a la entrada de la red (el ciclo) y z_t a la salida que se dirige a la siguiente capa. En las entradas de la red recurrente clásica (izquierda) se reciben X_t , que es el propio input, y h_{t-1} , que equivale a la salida del ciclo. Por el contrario, la neurona LSTM fusiona h_t con z_t y toda esa información se vuelve a recibir como input en h_{t-1} . Se ha mencionado anteriormente que las LSTM regulan la información utilizando puertas (gates). Definimos cada una de ellas y su funcionalidad a continuación:

- Input Gate (puerta de entrada): escoge qué información entra a la red y cuál guarda a futuro
- Forget Gate (puerta del olvido): escoge qué información olvida y cuál mantiene
- Output Gate (puerta de salida): escoge qué información manda a la siguiente capa

En la figura superior aparece una cuarta puerta (Input Modulation Gate) que generalmente se incluye dentro de la puerta de entrada. Ahí se ha reflejado de forma explícita para que se entienda que hay un sistema regulador de cómo fluye la información nueva dentro de la neurona. En resumen, las neuronas LSTM aprenden cuándo permitir que entre información nueva, cuándo olvidarla o eliminarla. Esto se aprende a través de cada iteración, según se va

analizando el error y su propagación y se van ajustando los pesos mediante el algoritmo de descenso de gradiente.

Finalmente, se incluye una segunda figura similar a la Ilustración 10 pero que en este caso representa a la neurona LSTM deshaciendo el ciclo, en $t-1$, t y $t+1$. El motivo es que en esta Ilustración se aprecia mejor la combinación de funciones matemáticas y operaciones lineales que se llevan a cabo en una sola iteración. En la imagen previa aparecía un círculo llamado Cell que representaba el núcleo de la neurona. Aquí queda reconvertido a las dos funciones tangente hiperbólica (\tanh) en combinación con la operación lineal suma (+). Cada una de las puertas (input, output, forget) lleva asociada una función sigmoidal como se había comentado anteriormente. Esto consigue que sólo se propaguen valores en el rango $[0,1]$, lo cuál es útil para mantener intacta la información de origen identificada como positiva.

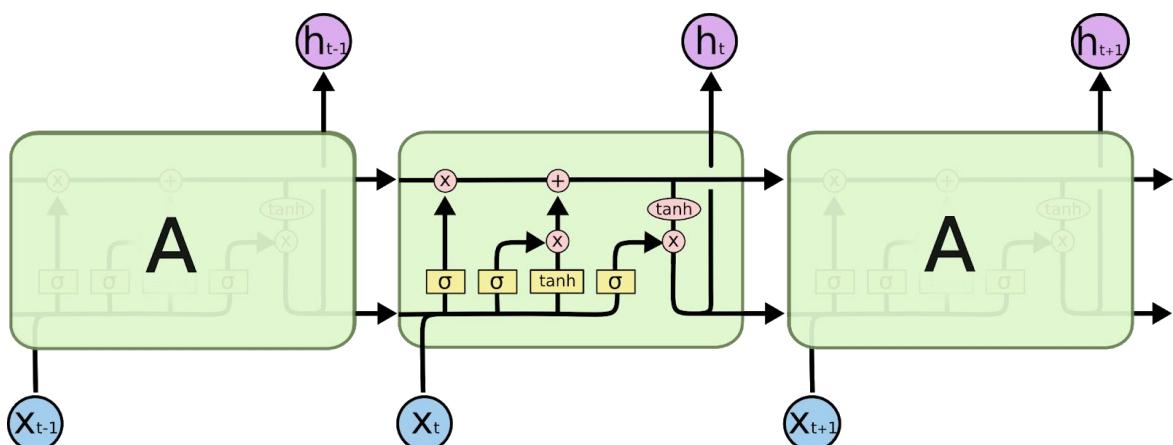


Ilustración 32: Neuronas de la red LSTM

La primera columna representa la puerta input, la segunda la puerta forget y la tercera la puerta output. La operación suma de input y forget permite regular la cantidad de información que se almacena y la que se propaga. La función tangente hiperbólica trabaja en el intervalo $[-1,1]$ y siempre aparece al lado de una sigmoide. Esto es para nivelar la información que propaga la sigmoide y escoger sólo la que verdaderamente nos hace falta. Por ejemplo, una neurona podría encargarse exclusivamente de analizar si la forma grammatical es singular o plural.

Para visualizar ambas funciones, se acompaña la explicación de la siguiente figura. Si se desea ampliar el conocimiento a nivel teórico sobre las redes LSTM se recomienda la lectura [17].

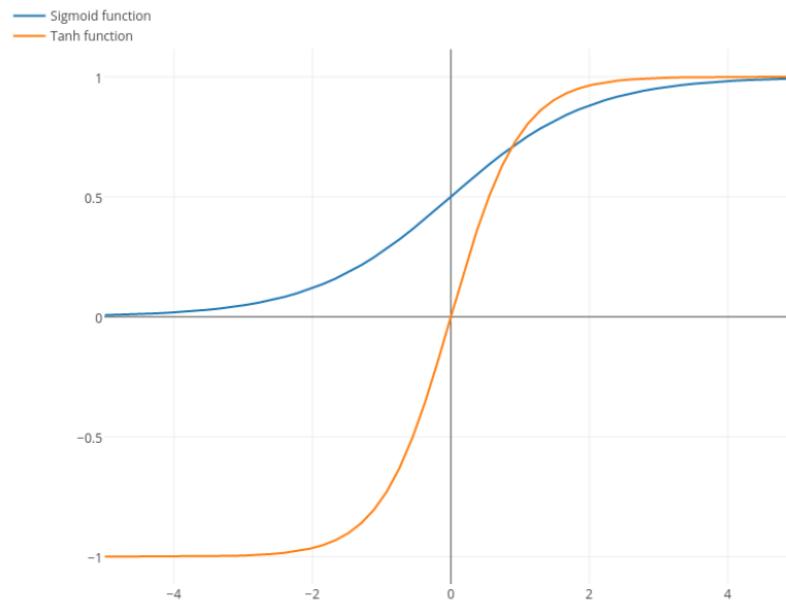


Ilustración 33: Función sigmoidal y tangente hiperbólica

7.4. Redes LSTM en combinación con redes convolucionales

Para cerrar este tema, se pretende unir lo aprendido en Computer Vision con esta parte nueva de Natural Language Processing.

Con el objetivo de realizar tareas de aprendizaje sobre multitud de datos, es muy común encontrarse el problema de describir mediante lenguaje natural qué está sucediendo en una imagen. Ya estudiamos este tema en Computer Vision mediante el auto-tagging y el auto-labelling, pero no profundizamos en la tarea de NLP que radica en la descripción de la imagen.

Cuando esto sucede, lo más sencillo es combinar una red convolucional que extraiga las características de la imagen con un conjunto de redes LSTM que, utilizando dichas características, las relacione con palabras de un determinado idioma que describan la imagen recibida como entrada. En la figura 14 se puede ver la estructura que adoptaría la red para reconocer una imagen en la que aparece un grupo de jirafas.

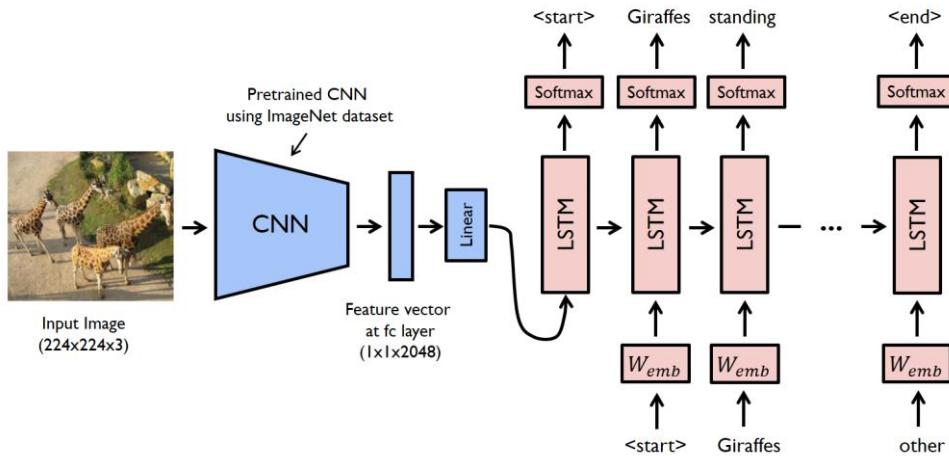


Ilustración 34: Red convolucional en combinación con red LSTM

8. REFERENCIAS

- [1] CIFAR-10 and CIFAR-100 dataset <https://www.cs.toronto.edu/~kriz/cifar.html>
- [2] Image-net dataset <http://www.image-net.org/>
- [3] Google Cloud Vision <https://cloud.google.com/vision/>
- [4] Google Cloud Vision for developers <https://cloud.google.com/vision/docs/tutorials>
- [5] IBM Watson Visual Recognition <https://www.ibm.com/watson/services/visual-recognition/demo/#demo>
- [6] Microsoft Azure Cognitive Services <https://azure.microsoft.com/es-es/services/cognitive-services/computer-vision/>
- [7] Tensor Flow <https://www.tensorflow.org/>
- [8] Language pitch analysis <https://erikbern.com/2017/02/01/language-pitch.html>
- [9] M.F Porter (1980) An algorithm for suffix stripping <http://stp.lingfil.uu.se/~marie/undervisning/textanalys16/porter.pdf>
- [10] Tutorial sobre text-mining en R <https://www.tidytextmining.com/index.html>
- [11] AFINN dataset http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010
- [12] Bing dataset <https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#lexicon>
- [13] NRC dataset <http://saifmohammad.com/WebPages/NRC-Emotion-Lexicon.htm>
- [14] Dataset recap for NLP <https://gengo.ai/datasets/the-best-25-datasets-for-natural-language-processing/>
- [15] IBM Watson <https://www.ibm.com/watson/products-services/>
- [16] F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count (2000). *IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, Como, Italy, 2000, pp. 189-194 vol.3. doi: 10.1109/IJCNN.2000.861302 ftp://ftp.idsia.ch/pub/juergen/TimeCount-IJCNN2000.pdf

[17] LSTM networks <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>