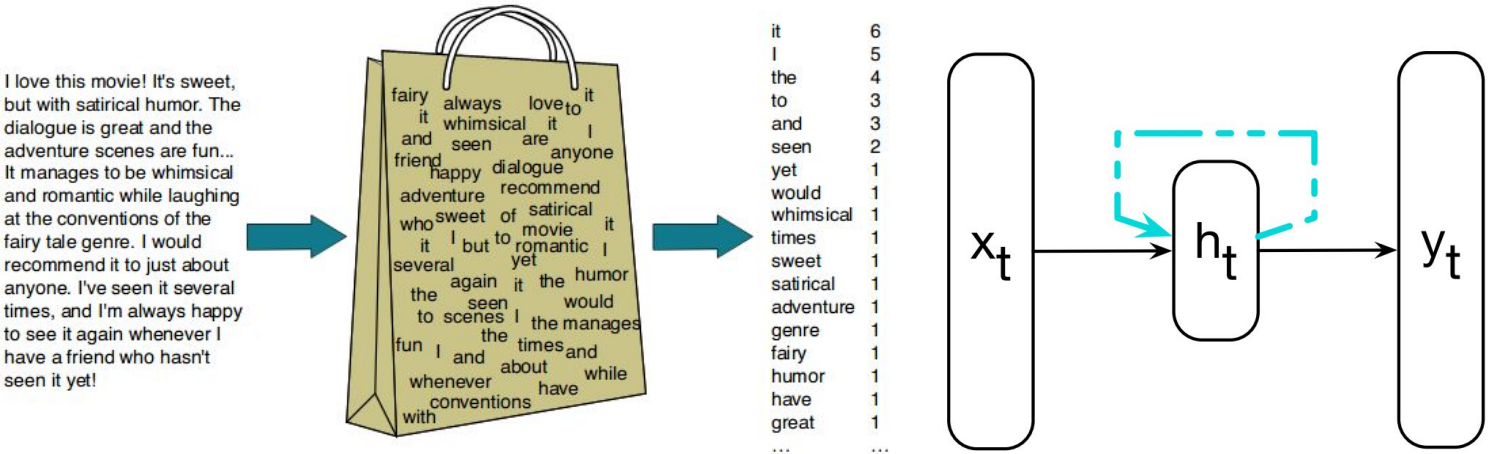
Transformers

Fabián Villena

Representaciones de texto

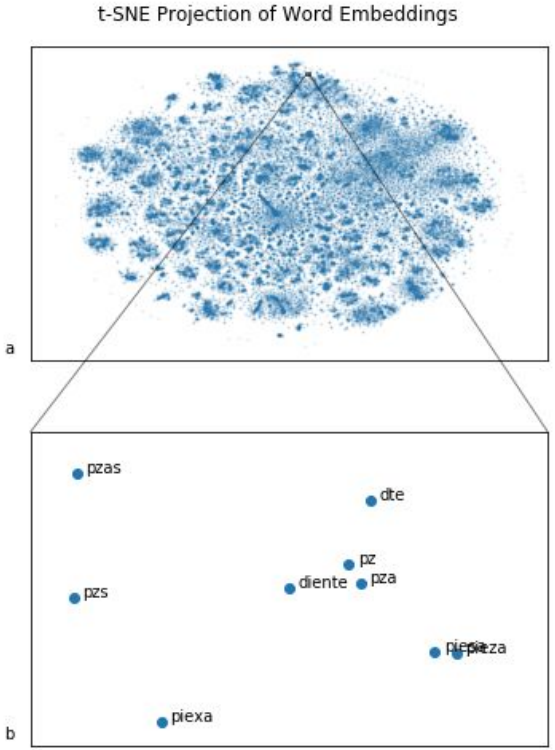
Para poder representar un texto podemos utilizar **métodos simples de vectorización que no toman en cuenta el orden de las palabras** dentro del documento (bolsa de palabras o TF*IDF) o podemos utilizar representaciones más complejas que pueden tomar en cuenta el orden de las palabras.



Word embeddings

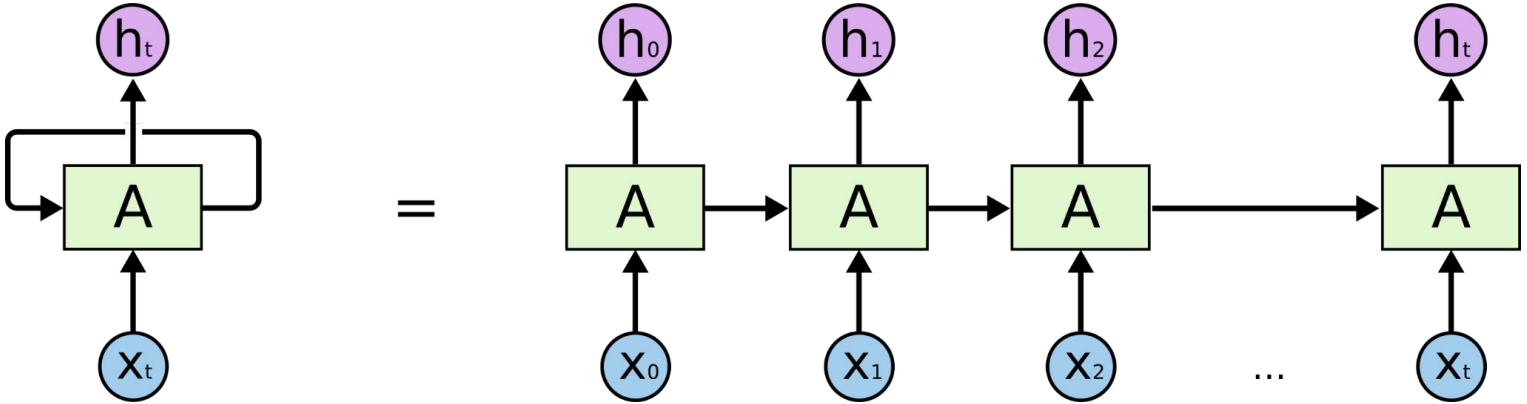
Las representaciones de texto modernas aprenden **representaciones unitarias de cada una de las palabras del conjunto de entrenamiento** para posteriormente poder representar una secuencia de texto.

Estas representaciones independientes de palabras pueden ser mejoradas al generar representaciones que tomen en cuenta el contexto para determinar la representación de las palabras.



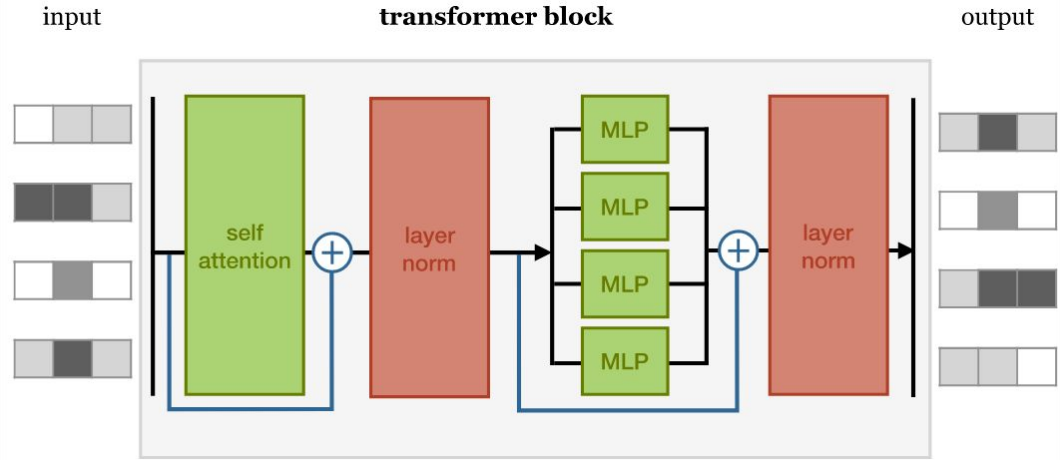
Redes Neuronales Recurrentes

Una de las formas modernas de representación de texto que toma en cuenta el orden de las palabras es **la utilización de redes neuronales recurrentes para que desde los embeddings de las palabras, podemos obtener una representación de un documento completo**. El problema es que estas representaciones son seriales, por ende su cálculo no es eficiente y además se pierde la información distante.



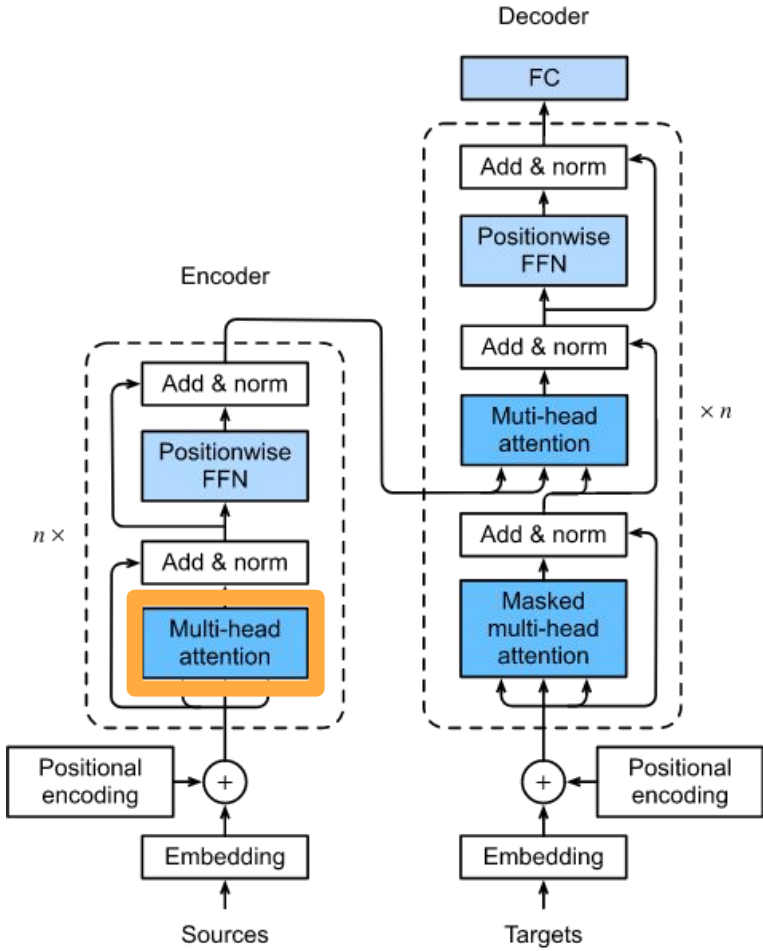
Transformers

El problema de la información distante y la naturaleza secuencial de las redes recurrentes llevaron al desarrollo de los Transformers; un acercamiento al **procesamiento de secuencias que elimina las conexiones recurrentes** y se asemeja más a las redes totalmente conectadas.



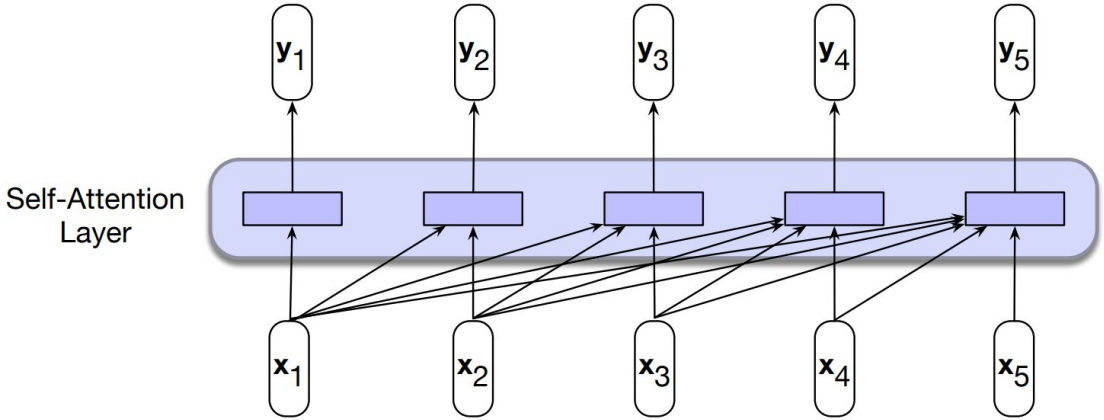
Arquitectura de los Transformers

Estas arquitecturas mapean secuencias de vectores de entrada a secuencias de vectores de salida de las mismas dimensiones. **Los Transformers se componen por una pila de *transformer blocks*, las cuales son redes multicapa** conformadas por la combinación de simples capas lineales, redes totalmente conectadas y capas de autoatención, donde esto último es la gran innovación de estas arquitecturas.



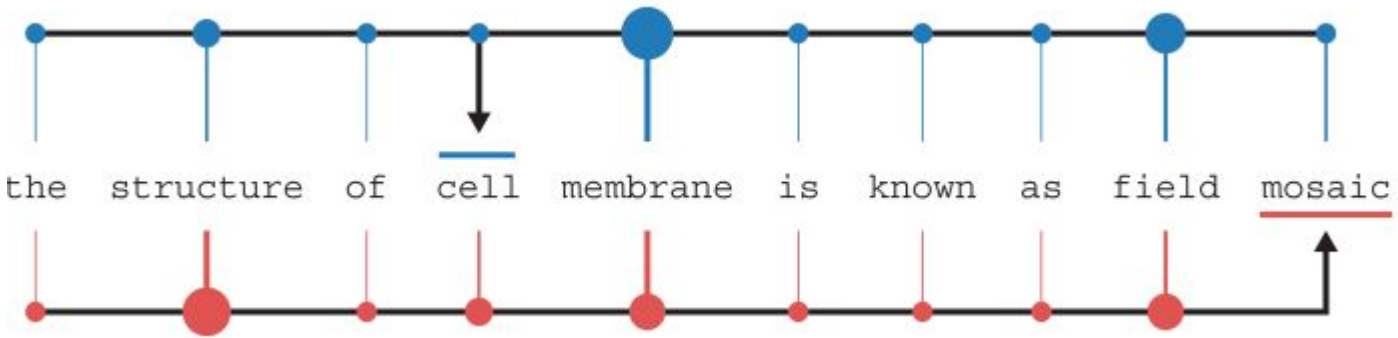
Autoatención

La **autoatención permite a la red directamente extraer y usar información desde contextos arbitrariamente largos sin la necesidad de pasar a través de conexiones recurrentes intermedias**. Como un transformer en general, las capas de autoatención mapean secuencias de entrada hacia secuencias de salida del mismo tamaño.



La autoatención contextualiza las palabras

Con la autoatención podemos **mezclar los vectores de entrada individuales para contextualizar la información de cada palabra aislada**. La salida de esta capa genera la misma cantidad de vectores de entrada, pero estos son mezclas de los vectores de entrada en función de su relevancia para la el vector de entrada.

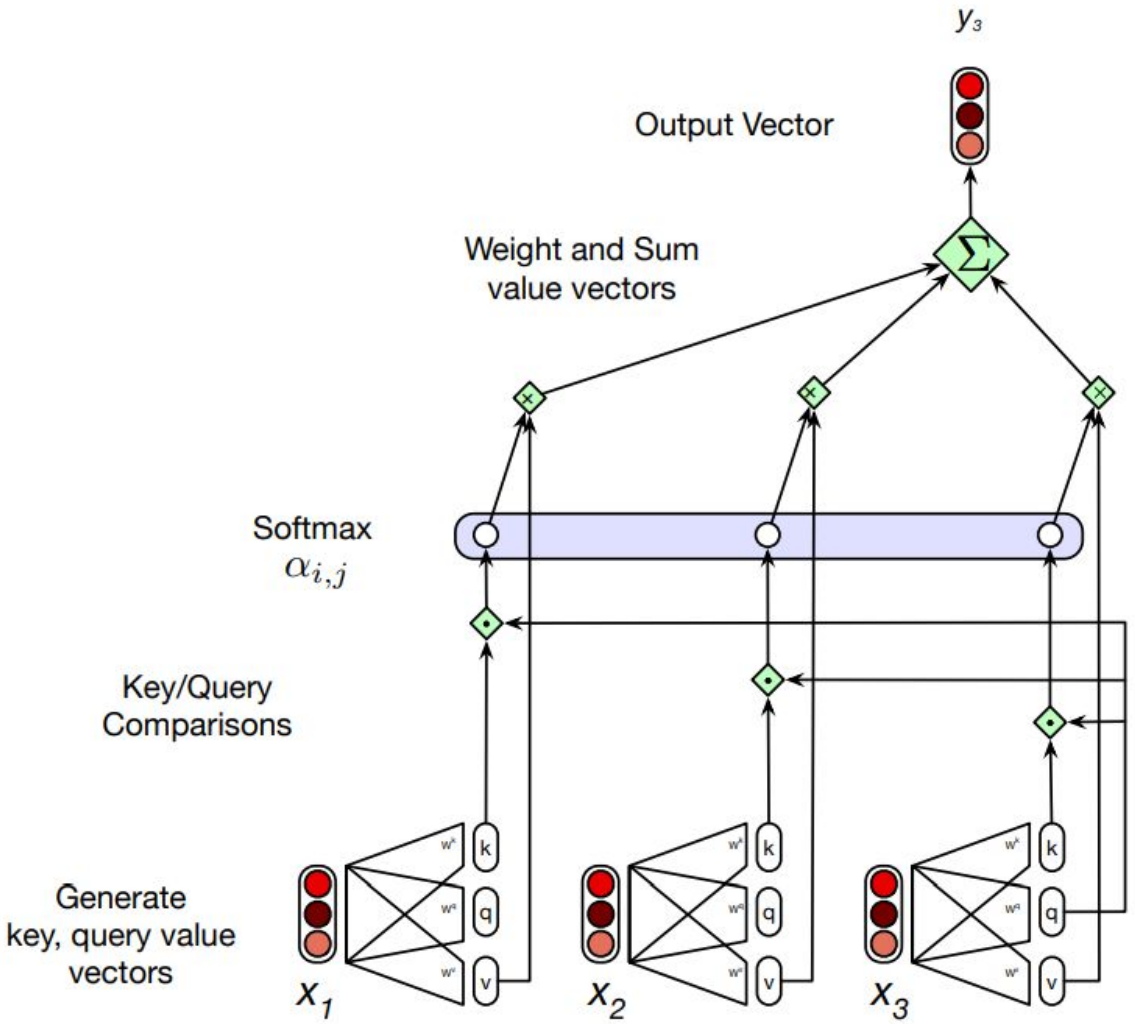


Algoritmo de autoatención

1. Para cada entrada x_i calcular tres diferentes vectores: *query* q_i, *key*, k_i y *value* v_i al multiplicar x_i por por tres matrices, W^q, W^i y W^v.
2. Calcular un *score* s_i por cada entrada x_i al realizar producto punto entre q_i y todos los k_i.
3. Normalizar los scores.
4. Calcular un promedio ponderado de cada v_i, ponderado por su correspondiente score s_i.

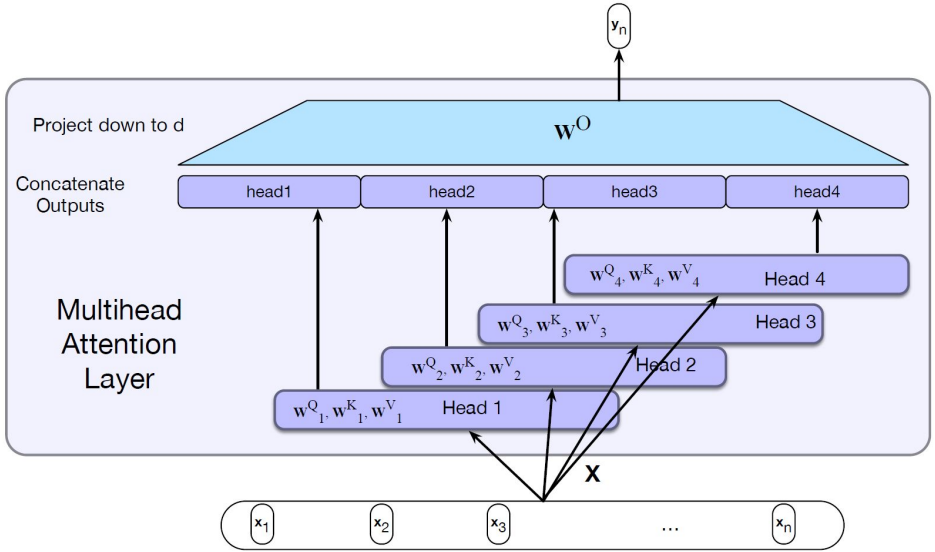
Autoatención

El vector de salida es una combinación de cada uno de los vectores de entrada, en donde **se pondera de mayor manera a los vectores que son más relevantes como contexto**.



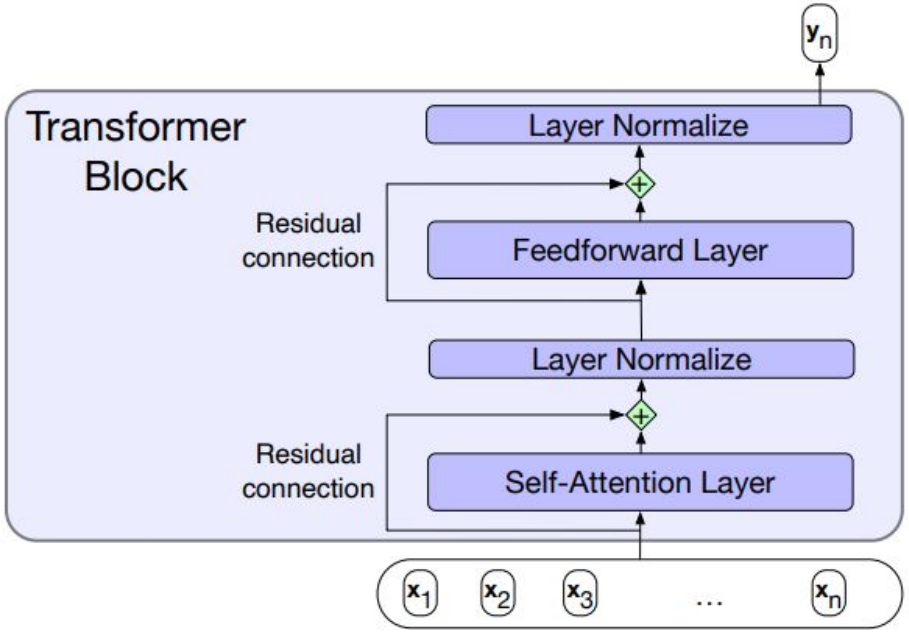
Atención multicabezal

Las distintas palabras en una oración se pueden relacionar con otras palabras de distintas maneras. Un sólo *transformer block* no sería suficiente para extraer todos los tipos de relaciones que existen entre palabras. **La atención multicabezal son múltiples capas paralelas de autoatención**, llamadas *heads*, que contienen sus propios conjuntos de parámetros.



Transformer block

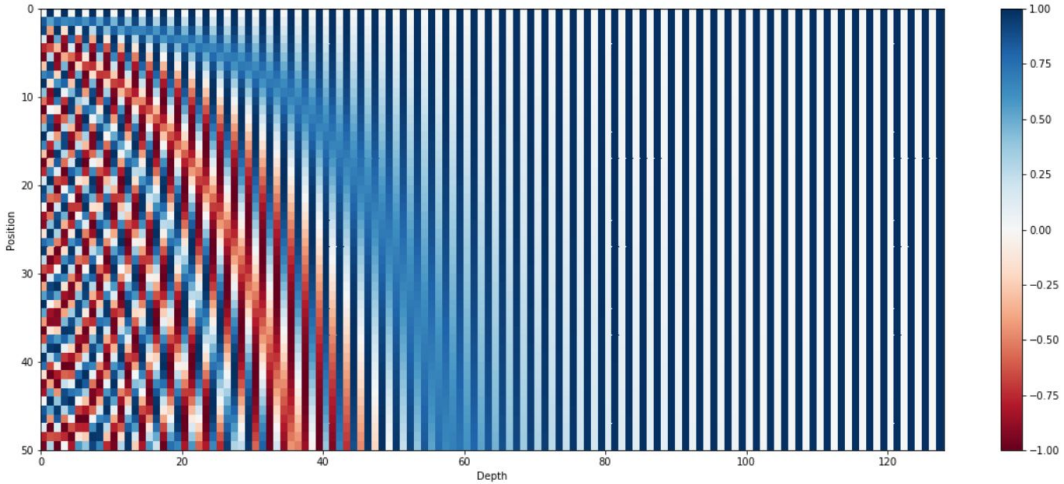
La autoatención es el centro de los bloques de transformers, estos bloques pueden contener más elementos, como redes densas y normalizaciones, junto con conexiones residuales que mejoran el rendimiento de las redes.



Codificador posicional

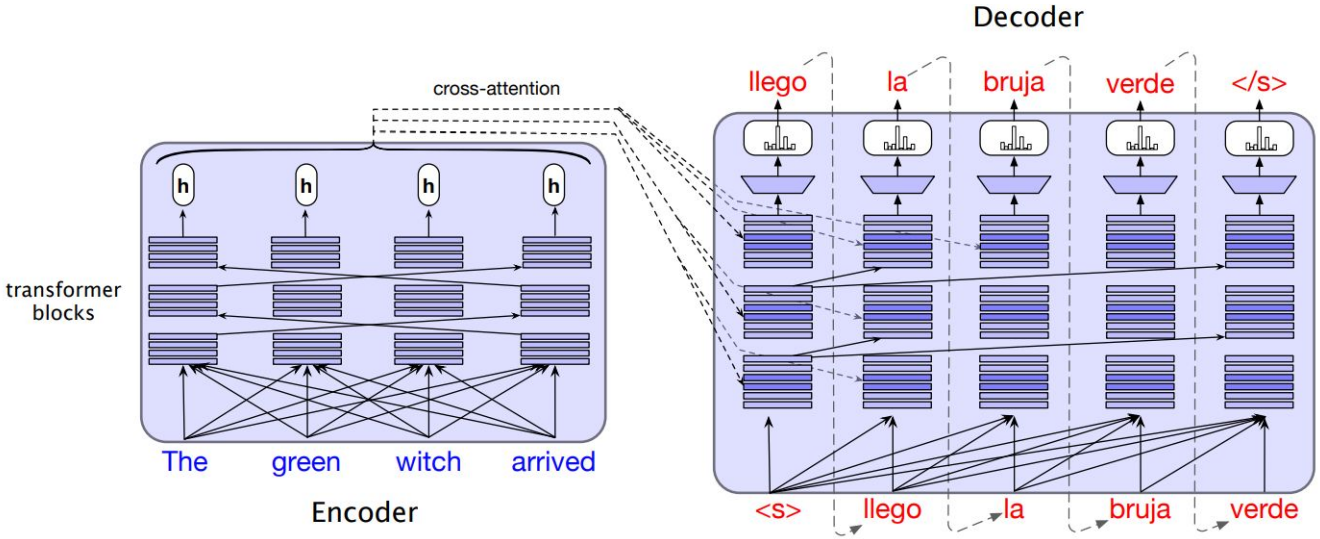
Al contrario de las redes recurrentes, estas arquitecturas no tienen la noción de posición de las palabras incluida en si misma.

Para representar la posición de las palabras, una combinación de funciones seno y coseno con distintas frecuencias son utilizadas.



Encoder-Decoder

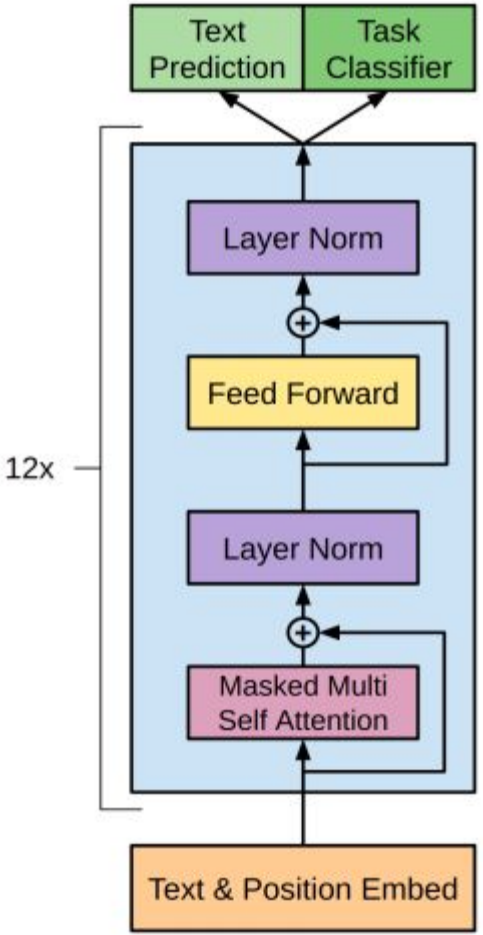
La estructura encoder-decoder también se puede implementar con transformers. Esto consiste en un encoder que representa la secuencia de entrada y el decoder es un modelo de lenguaje condicional que utiliza la secuencia codificada para generar las palabras de la secuencia de salida.



GPT

Generative Pretrained Transformer es una arquitectura que **utiliza sólo un decoder para ajustar un modelo de lenguaje para la predicción de la siguiente palabra**.

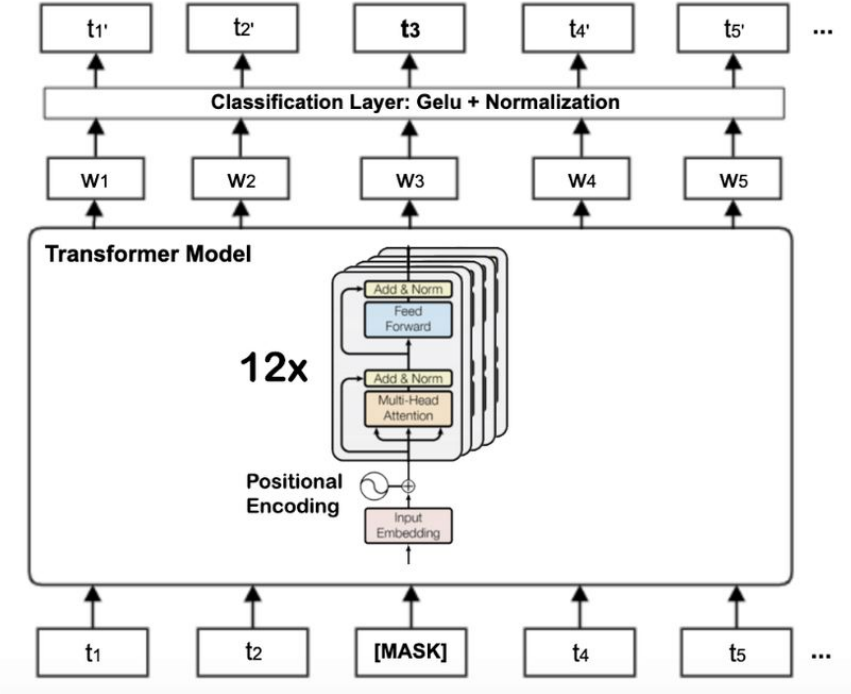
Esta arquitectura sólo puede atender a las palabras previamente vistas para predecir la siguiente. Esto puede ser limitante para poder codificar oraciones, porque uno debe observar toda la oración para poder entender una palabra.



BERT

Bidirectional Encoder Representations from Transformers utiliza la arquitectura de encoder.

Esta arquitectura permite **obtener información de la secuencia completa a través de cambiar la tarea de predicción de la siguiente palabra por la predicción de una palabra enmascarada** dentro de la oración.



BETO

Este modelo de la arquitectura BERT **fue entrenado con el Big Spanish Corpus** utilizando con la técnica de modelado de lenguaje enmascarado.

Este modelo contiene un vocabulario de 31 mil subpalabras calculadas utilizando SentencePiece.

https://huggingface.co/dccuchile/bert-base-spanish-wwm-uncased

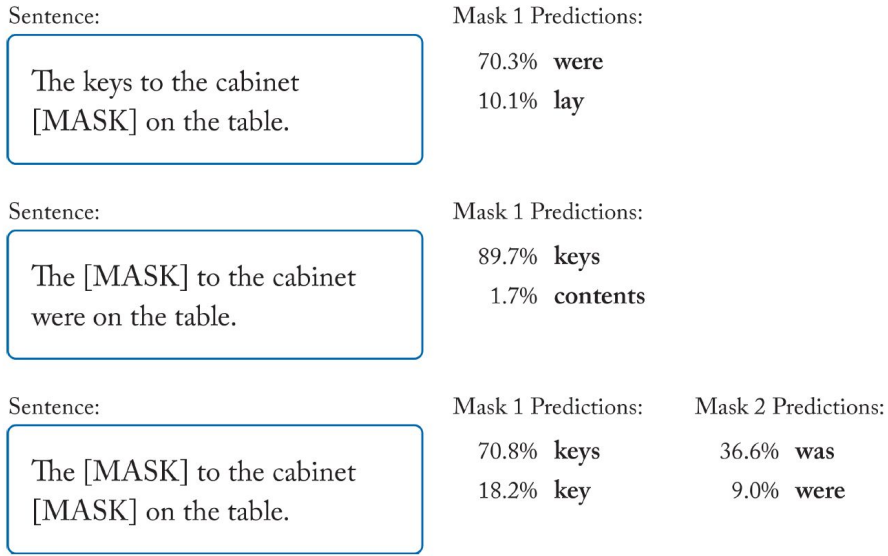
Task	BETO-cased	BETO-uncased	Best Multilingual BERT
POS	**98.97**	98.44	97.10 [2]
NER-C	**88.43**	82.67	87.38 [2]
MLDoc	95.60	**96.12**	95.70 [2]
PAWS-X	89.05	89.55	90.70 [8]
XNLI	**82.01**	80.15	78.50 [2]

Modelado de lenguaje enmascarado

Anteriormente el objetivo del modelamiento de lenguaje era la predicción de la siguiente palabra.

Este nuevo modelamiento de lenguaje **tiene el objetivo de predecir una palabra enmascarada dentro de una oración**.

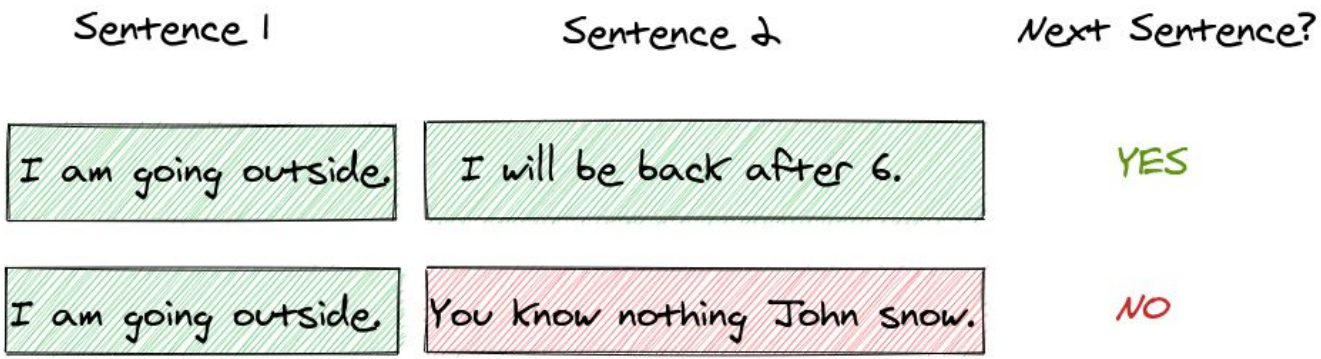
Esto permite que el modelo no se condiciona sólo por el contexto izquierdo o derecho, sino que por ambos lados.



Predicción de la siguiente oración

Además de la predicción de palabras enmascaradas, BERT predice si una oración es posterior a la oración actual o no.

Esto es **motivado porque hay tareas en donde el modelo necesita codificar relaciones entre oraciones** o extraer información fuera de los límites de una oración.



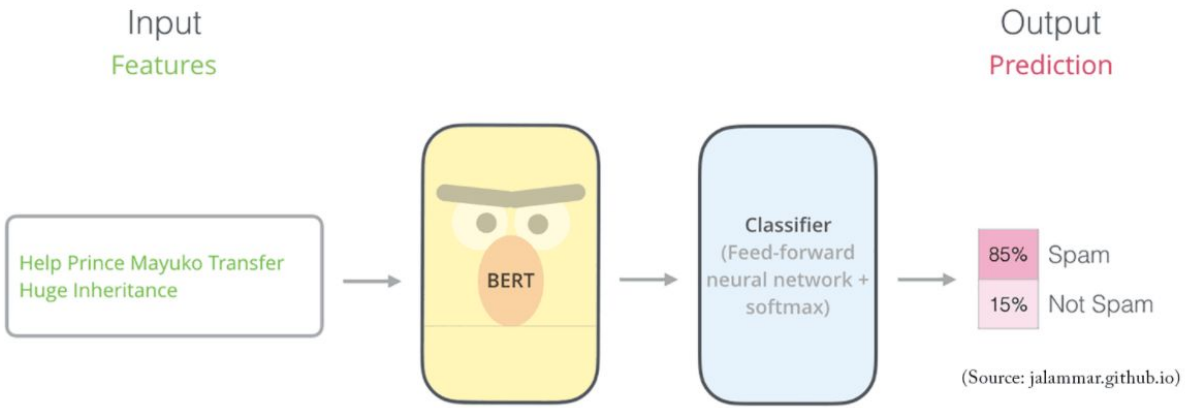
Entrenamiento

El entrenamiento de BERT se basa en la **minimización de una función de pérdida combinada entre la tarea de modelado de lenguaje enmascarado y la tarea de predicción de la siguiente oración**.

Originalmente BERT es entrenado en dos versiones, una *base* y una *large*, las cuales difieren en la cantidad de capas encoder, tamaño de la representación y el número de heads.

Afinamiento

El afinamiento (*fine tuning*) de un modelo significa **utilizar un modelo de lenguaje pre entrenado para resolver una tarea específica**. Normalmente lo que se realiza es simplemente añadir capas al final del modelo de lenguaje que puedan resolver la tarea que deseamos.



Grandes modelos de lenguaje

Los grandes modelos de lenguaje son modelos pre entrenados basados en Transformers que tienen órdenes de magnitud más parámetros que los modelos pre entrenados anteriores.

Por ejemplo el modelo pre entrenado BERT tiene $0.3*10^9$ parámetros y GPT-3 tiene $175*10^9$ parámetros.

Se sabe que escalar el tamaño de los modelos de lenguaje mejora el rendimiento de los modelos en tareas específicas.

Habilidades emergentes

Las habilidades emergentes son aptitudes que no están presentes en los modelos de lenguaje pequeños, pero que sí aparecen en los grandes modelos de lenguaje:

- In-Context Learning: Un modelo puede generar salidas esperadas a instrucciones dadas en lenguaje natural sin entrenamiento adicional
- Instruction Following: Un modelo funciona bien en tareas nunca antes vistas que también son descritas utilizando instrucciones en lenguaje natural
- Step-by-step reasoning: Un modelo puede resolver tareas complejas al instruir el modelo con los razonamientos intermedios para llegar a la respuesta final.

ChatGPT se sacó un 7 en el control

https://chat.openai.com/share/47dae801-9467-4197-9266-f76f689d07c8

AI explotar la habilidad emergente de In-context learning, ChatGPT puede adaptarse y resolver tareas instruidas en lenguaje natural sin haber sido entrenado para esa tarea específica.