# Grandes Modelos de Lenguaje

Fabián Villena

# Introducción

La utilización de modelos de lenguaje basados en Transformers ha revolucionado el campo del NLP debido a su modularidad para ser adaptado a múltiples tareas, pero se sabe que el tamaño del modelo afecta su rendimiento.

Los grandes modelos de lenguaje tienen órdenes de magnitud más parámetros que los modelos pre entrenados anteriores y se ha observado que presentan habilidades extraordinarias.

# Modelos basados en 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.

Todos los modelos del estado del arte están basados en la arquitectura de red neuronal de Transformers.

# Modelos pre entrenados pequeños vs. grandes

Los modelos preentrenados sobre grandes cantidades de texto como BERT, RoBERTa o DeBERTa han sido el estándar para resolver tareas de NLP ultimamente.

Estos modelos muestran buenos resultados, pero su tamaño afecta su capacidad, por ejemplo BERT tiene 0.3e9 parámetros y el gran modelo de lenguaje GPT-3 tiene 175e9 parámetros. Se ha encontrado que escalando los modelos preentrenados se mejora el rendimiento de los modelos.

# Habilidades emergentes

Además de un rendimiento superior en tareas de NLP, los grandes modelos de lenguaje muestran otros comportamientos importantes y sorprendentes en la resolución de tareas complejas, llamadas habilidades emergentes.

Estas habilidades emergentes no están presentes en pequeños modelos de lenguaje, pero sí aparecen en grandes modelos de lenguaje.

Unas de las habilidades emergentes son el aprendizaje en contexto, el seguimiento de instrucciones y el razonamiento paso a paso.

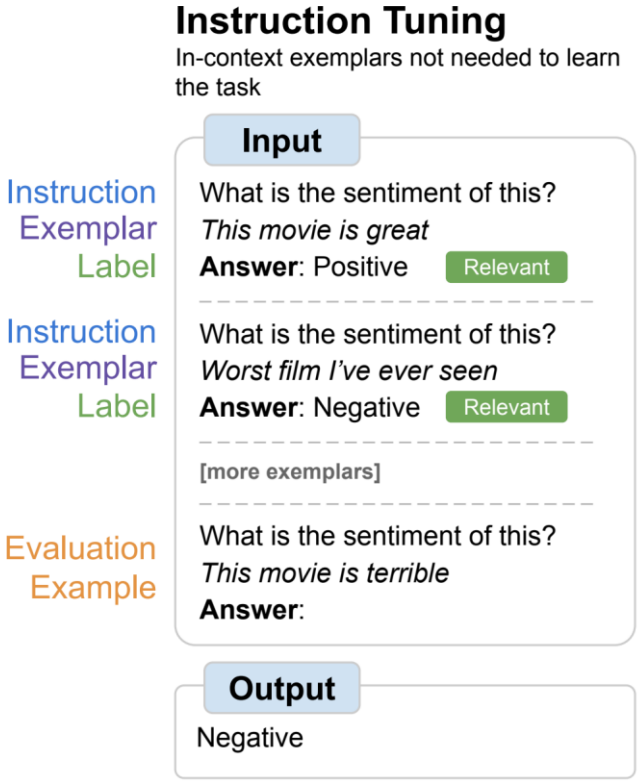
# Aprendizaje en contexto

El aprendizaje en contexto es un paradigma que permite a los modelos de lenguaje aprender tareas sólo al darle unos cuantos ejemplos en forma de demostración.

A pesar que los grandes modelos de lenguaje en sí presentan buenas capacidades de aprendizaje en contexto, algunas técnicas como el afinamiento supervisado de instrucciones se utiliza para mejorar las capacidades de aprendizaje en contexto al ajustar los parámetros del modelo al entrenarlo en tareas de seguimiento de instrucciones.

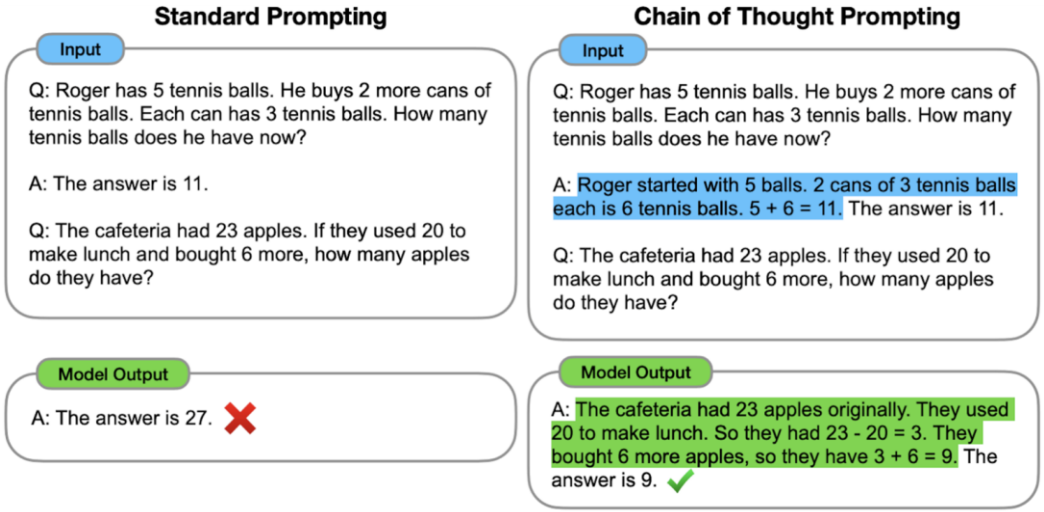
# Seguimiento de instrucciones

Esta habilidad emergente se refiere a la capacidad de un modelo entrenado para el seguimiento de instrucciones se comporta bien en tareas no antes vistas que también están descritas en forma de instrucciones.



# Razonamiento paso a paso

Esta habilidad emergente se refiere a que un modelo puede resolver problemas complejos a instruir al modelo utilizando razonamientos intermedios para derivar la respuesta final.



# Prompting

En NLP tradicional tomamos una entrada de texto y predecimos una salida basada en un objeto que modela la probabilidad condicional de que esa entrada de texto pertenezca a una clase dado un conjunto de parámetros aprendidos. Para aprender esos parámetros utilizamos un conjunto de datos que tiene pares de entrada y salida y entrenamos el modelo para predecir esa probabilidad condicional.

Los modelos de NLP basados en prompting intentan evitar la fase de entrenamiento tarea-específica al modelar la probabilidad de una secuencia de texto exista dado un conjunto de parámetros y usar esa probabilidad para predecir la salida.

# Adición del prompt

En el primer paso del prompting una función es aplicada para modificar el texto de entrada hacia un prompt. Primero se aplica una plantilla de texto que tiene dos casilleros, uno para el texto de entrada y otro para la respuesta y después se llena el casillero del texto de entrada.

[X]. En general la película me pareció [Z].

[X] = *Amé esa película*

*Amé esa película*. En general la película me pareció [Z].

# Búsqueda de la respuesta

Después, buscamos el texto [Z] que maximice el puntaje modelado por el modelo de lenguaje. La función de búsqueda podría ser por ejemplo `argmax`.

*Amé esa película*. En general la película me pareció [Z].

[Z] = {buena, mala, terrible, bacán}

Amé esa película. En general la película me pareció *bacán*.

# Mapeo de la respuesta

Finalmente debemos pasar desde la respuesta que obtuvo el mayor puntaje hacia el espacio de etiquetas que satisface nuestra tarea. A veces esto es trivial pero hay casos en donde múltiples respuestas pueden resultar en el mismo valor de salida, por ejemplo uno puede usar distintas palabras para referirse al mismo sentimiento, por lo tanto sería necesario tener un mapeo entre las respuestas encontradas y valores de salida.

\+ = {buena, bacán}

\- = {mala, horrible}

# Consideraciones de diseño del prompt

- Selección del modelo de lenguaje pre entrenado
  - Debemos elegir un modelo de lenguaje que sea lo más cercano a nuestro dominio.
- Ingeniería de la plantilla del prompt
  - Dado que la plantilla determina la tarea, la elección de la plantilla de prompt adecuada tiene un gran efecto en el rendimiento.
- Ingeniería de la respuesta del prompt
  - Dependiendo de la tarea, se necesita diseñar la respuesta de manera acorde, junto con la función de mapeo de la respuesta.

# Fine-tuning de instrucciones

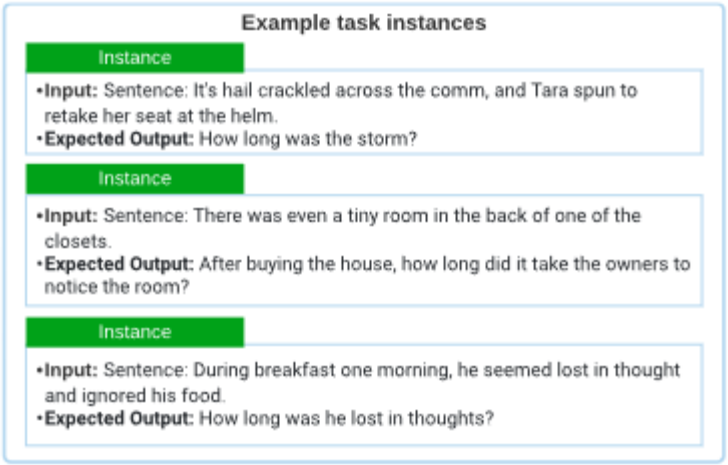
Los grandes modelos de lenguaje son típicamente entrenados para minimizar el error contextual de predicción de palabras, pero los usuarios desean que el modelo específicamente siga instrucciones.

El fine-tuning de instrucciones es una técnica efectiva para mejorar las capacidades y controlabilidad de los grandes modelos de lenguaje.

# Conjuntos de datos de instrucciones

Los conjuntos de datos de instrucciones incluyen instrucciones y salidas, en donde la instrucción denota la instrucción humana que se le da al modelo y la salida denota la salida deseada que debiese seguir posterior a la instrucción.

Cada instancia en un conjunto de datos de instrucciones consiste en una instrucción, una entrada opcional como contexto y una salida dada la instrucción y la entrada.



# Plantilla de instrucciones

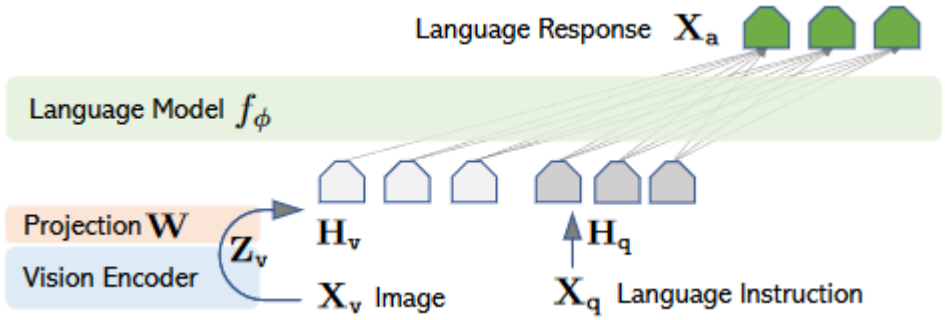
Plantilla de instrucciones de Llama 2:

```
<s>[INST] <<SYS>>
{your_system_message}
<</SYS>>

{user_message_1} [/INST] {model_reply_1}</s><s>[INST]
{user_message_2} [/INST]
```

# Multimodalidad

Se representa una imagen a través de un visual encoder y se conecta con el espacio de los embeddings de texto a través de una capa lineal obteniendo una secuencia de tokens visuales, después se utiliza este secuencia de tokens para predecir una respuesta de texto.



# Generación de la respuesta

Los grandes modelos de lenguaje están entrenados con el objetivo de predecir la probabilidad de una secuencia de palabras y típicamente se utilizan para predecir la siguiente palabra en una oración de contexto.

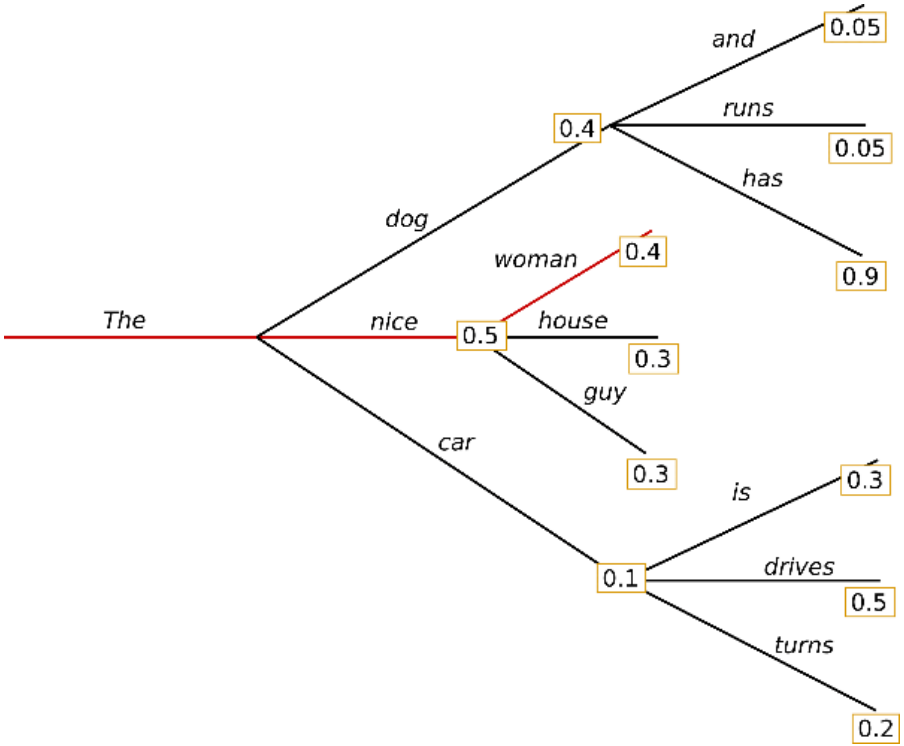
Existen múltiples formas de generar texto desde esta distribución de probabilidad de palabras.

# Greedy search

Con esta estrategia de decodificación en cada paso temporal se predice la palabra que tiene la mayor probabilidad de aparecer como palabra siguiente.

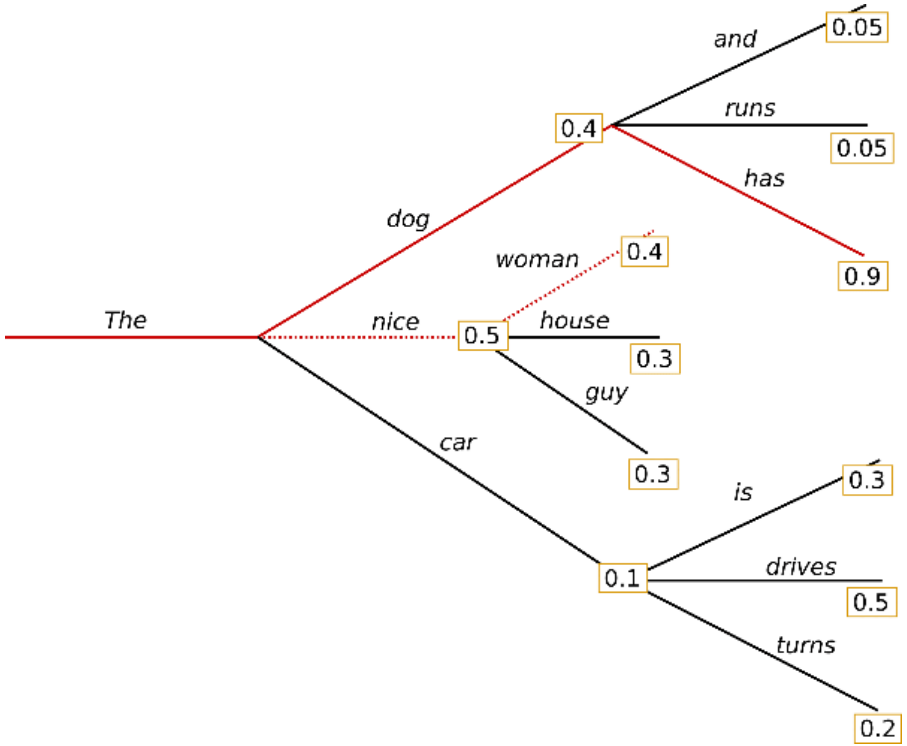
El problema con esta estrategia de decodificación es que las secuencias comienzan a repetirse.

El problema de esta estrategia es que palabras con alta probabilidad están escondidas tras palabras de baja probabilidad



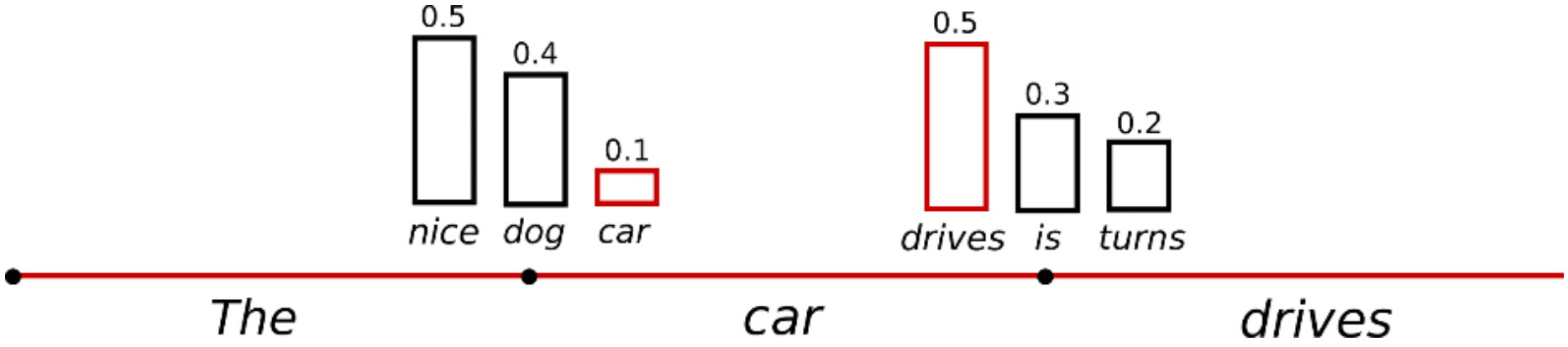
# Beam search

Con la estrategia de beam search se predice el riesgo de perder palabras con alta probabilidad a través de ir manteniendo una cantidad de hipótesis de beams en cada paso temporal y eventualmente eligiendo la hipótesis que tiene la mayor probabilidad en general.



# Muestreo

En su forma más básica, el muestreo significa tomar al azar una palabra siguiente de acuerdo a su distribución de probabilidad condicional. La decodificación ya no es determinista. El problema con esta decodificación es que genera texto que no parece ser generado por un humano porque existe la posibilidad de elegir palabras con poca probabilidad.



# Temperatura

Un truco para generar texto más coherente es modificar la distribución de probabilidad haciéndola más marcada. Aumentando la probabilidad de palabras más probables y disminuyendo la probabilidad de palabras menos probables

$$p(x_i) = \frac{e^{\frac{x_i}{T}}}{\sum_{j=1}^{V} e^{\frac{x_j}{T}}}$$