Estrategias de Decodificación

Andrea Manuel Simón y Ana López Palomo

October 2023

1 Descripción Detallada de LLM

Hay una idea errónea común de que los LLM como GPT-2 producen texto directamente. Este no es el caso. En cambio, los LLM calculan logits, que son un conjunto de números que representan la probabilidad de cada palabra en el vocabulario, como un ID. Durante el proceso de generación de texto, el modelo se mueve desde el primer logit hasta el último, y para cada lugar en el texto se escoge, mediante un algoritmo, la palabra (es decir, el token) más adecuada.

Tras escoger todos los logits, se aplica una función de activación, que hace que cada token se convierta en una probabilidad normalizada. Luego, se selecciona la palabra con la probabilidad más alta, teniendo en cuenta las variaciones de decodificación que explicamos más adelante.

De esta manera, GPT, forma una frase, pero no la "inventa". GPT es autorregresivo, esto quiere decir que predice la siguiente palabra en base a todos los tokens anteriores, multiplicando las probabilidades de la secuencia de tokens.

2 Proceso de Generación de Texto

Los LLM han sido entrenados para entender y generar grandes cantidades de texto. Pasos:

- Se entrena el modelo. Este modelo es un gran corpus de texto con el objetivo de aprender los patrones y las relaciones entre palabras. Todo ello mediante técnicas de aprendizaje automático para optimizar la capacidad del modelo de texto.

- Codificar-decodificar. Primero realiza una codificación, es decir, procesa la entrada de texto (la pregunta introducida) y genera un código de representación. El procesamiento del código se realiza mediante Red Recurrent, un tipo de red neuroral que procesa la pregunta letra por letra. Después hace la decodificación, que genera una secuencia de palabras coherentes en base a al pregunta original. La secuencia de palabras se hace una a una utilizando la predicción de la siguiente, basándose en el contexto anterior.

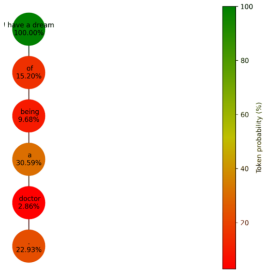
- Revisión. Un humano revisa y filtra el resultado para que sea coherente y relevante para la pregunta original. Por supuesto, se puede reentrenar el modelo basándonos en la respuesta.

Para enterder este procedimiento vamos a verlo con un ejemplo: Supongamos que el texto de entrada es: "Hola buenos días, ¿qué tal estás?". Tras recibirlo y procesarlo, lo divide en tokens almacenados en una lista: ["Hola", "buenos", "días", ",", "¿", "qué", "tal", "estás", "?"], donde cada uno le corresponde un ID: ["ID1", "ID2", "ID3", "ID4", "ID5", "ID6", "ID7", "ID8", "ID9"]. Con esto tenemos ya en ID, el proceso contrario sería de ID a tokens, donde reconstruiría la frase inicial.

3 Estrategias de Decodificación

3.1 Búsqueda voraz o codiciosa

Se trata de la búsqueda del token más probable que pueda aparecer a continuación, sin considerar el resultado final de la oración. Esto hace que sea una propiedad eficaz y muy rápida, dado que no hay necesidad de realizar múltiples secuencias. Aunque tiene un inconveniente, la posibilidad de haber conseguido una secuencia mejor.
Ejemplo: En la imagen podemos observar que la probabilidad de "of" es bas-



tante alta, pero no era la mejor opción, ya que "being" tiene una probabilidad baja de aparición.

3.2 Búsqueda de beams

La búsqueda de beams utiliza un sistema parecido: busca la mayor probabilidad de las palabras, pero busca los n tokens más probables, siendo n el número de tokens totales para completar la frase. Este número se puede predefinir, si no, la ración acabará con un token de fin de secuencia. Es decir, buscamos una probabilidad general más alta, en vez de mirar palabra por palabra.

Al realizar este método con la frase 'I have a dream' conseguimos calcular las puntuaciones de 63 tokens y obtenemos 25 secuencias posibles. Ahora solo debemos de estraer la mejor secuencia. Para ello identificaremos el nodo hoja

con la secuencia más alta y buscaremos el camino más corto desde la raiz hasta dicha hoja. Cada nodo de esa ruta contiene un token de la secuencia óptima. Obtenemos que la mejor solución es 'i have a dream. I have a dream'.

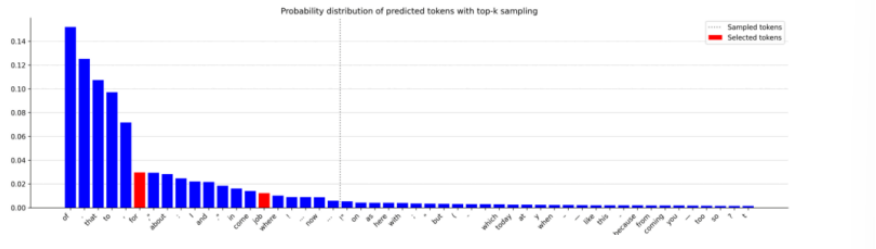
Podemos concluir que la solución de la búsqueda codiciosa da un resultado subóptimo, pero la búsqueda por beams tampoco nos proporciona un gran resultado.

3.3 Muestreo Top-K

Utiliza las probabilidades proporcionadas por el modelo de lenguaje y selecciona un token al azar entre los más probables. Para elegir la respuesta, aunque es al azar, utiliza un sistema de temperatura para cambiar las probabilidades de que salga cada opción.

Por ejemplo: en un caso que tengamos 4 opciones P(A)=20, P(B)= 10, P(C)=7, P(D)= 1

El algoritmo ignorará como solución la opción D, ya que es la menos probable, y escogerá una aleatoriamente entre las restantes. Aún así, con el uso de la temperatura, la probabilidad de que se escoja aleatoriamente A es más alta que B y así consecutivamente. Esto se usa normalmente para controlar el nivel de creatividad de la generación de texto, al ajustar la temperatura podemos generar respuestas más originales o predecibles.



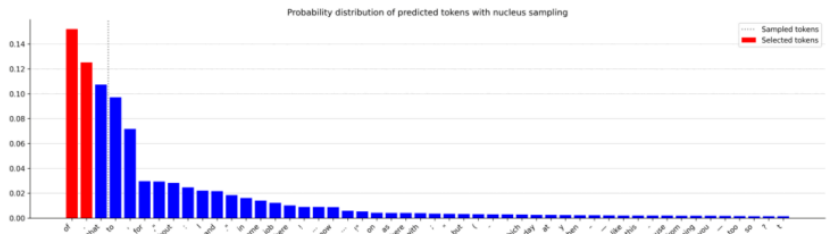
Con este gráfico podemos ver como funciona el método top-k. Muestra todos los tokens con más probabilidad de estar seleccionados a la izquierda y los tokens que tienen más probabilidadesde ser seleccionados la mayor parte del tiempo están marcados en rojo. Así podemos dirigir la oración hacia una menos predecible pero con más sentido.

El resultado obtenido poro GPT con este método es: "I have a dream job and I want to", una respuesta más natural que la del método de beams.

3.4 Muestreo del núcleo

EL muestreo del núcleo utiliza un sistema parecido al Top-k, pero para escoger entre las opciones con más probabilidad tiene una condición: la suma de las probabilidades de las palabras seleccionadas tiene que pasar un umbral, al cual

vamos a llamar P. Otra diferencia con el modelo anterior es que, de un paso a otro, la cantidad de tokens incluidos puede variar.



En este gráfico podemos ver que la cantidad de tokens en el núcleo influye bastante, dado que las distribuciones de probabilidad generadas varían haciendo que la selección de tokens no siempre esté entre los más probables.

La respuesta del algoritmo con este método es: "I have a dream. I'm going to".

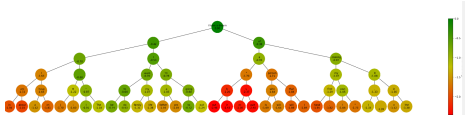


Figure 1: Enter Caption

A continuación podemos ver el árbol de decisión. Como podemos ver, este árbol es muy diferente al del muestreo codicioso, dado que contiene más variedad. Esto último nos lo proporcionan el muestreo top-k y el núcleo al ofrecernos más diversidad y creatividad en la respuesta.

4 Hiperparámetros y su Manipulación

Cuando se crea un modelo de lenguaje natural, hay varios hiperparámetros que se pueden ajustar para mejorar el rendimiento.

Los hiperparámetros son configuraciones, que pueden modificarse para mejorar la velocidad y la precisión de la generación del texto. Pueden incluir opciones como el tamaño del modelo, el tipo de red neuronal o la cantidad de nodos, entre otros.

Supongamos que queremos cambiar el tamaño del modelo, es decir, el número de parámetros y de nodos que componen la red neuronal. A medida que aumenta su tamaño, la capacidad de prodesamiento también aumenta, lo que puede resultar en una mejora en la calidad del texto.

La temperatura es un parámetro que controla, como hemos mencionado anteriormente, la variabilidad de laas respuestas generadas. Aumentando su

valor, podemos generar respuestas más variadas y creativas, mientras que si lo disminuimos se generarán respuestas más coherentes y precisas.

El número de beams es el número de decisiones qeu el modelo puede tomar en un momento dado durante la decodificación. Si aumentamos este parámetro, es más probable que el modelo encuentre la mejor opción, ya que estamos aumentando el número de secuencias alternativas. También aumentaría la diversidad de las respuestas, pero con un riesgo de sobregeneración. La sobregeneración se produce cuando el modelo devuelve muchas respuestas similares o redundantes.

Si cambiamos el parámetro top-k, podemos aumentar o disminuir el númro de palabras probables. De esta manera, podemos aumentar la diversidad de las respuestas, y evitar la repetición. Por otra parte, al disminuir este valor, podemos conseguir una mayor calidad de las respuesta, ya que el modelo considera solo las palabras más probables. El parámetro top-p, al ser similar al top-k, tiene los mismos efectos en el modelo.

5 Código y Práctica

El código viene adjunto en el archivo jupyter "Ejercicio".

6 Reflexión y Conclusiones

En conclusión, todas las distintas estrategias de decodificación que hemos visto tienen puntos a favor y puntos en contra. Por esto, dependiendo de lo que se necesite, es muy importante saber cual de todas usar.