Estrategias de Decodificación

jose Luis Rodríguez, Alexandre Muñoz y Gonzalo Martínez

October 2023

1 Descripción de LLMs

1.1 ¿Qué es un LLM?

Un LLM es esencialmente un sistema que tomando unas palabras como punto de partida (input) predice qué palabras son las más probables que ocurran a continuación (output). El sistema toma el input y calcula varias posibilidades asignando una probabilidad a cada una, y selecciona una en base a distintos criterios (su probabilidad, su conexión con otras palabras seleccionadas en el mismo texto...).

Los LLMs "aprenden" cuáles son estas probabilidades a partir de observar (entrenarse con) grandes cantidades de textos. Un LLM es solo una gigantesca y sofisticada máquina de calcular probabilidades. Pero es una máquina tan grande, que surgen comportamientos emergentes.

La calidad de los resultados de un LLM dependen de la cantidad y tipo de texto que hayan observado (con el que hayan sido entrenados): si el modelo no es muy grande, no ofrecerá buenos resultados. Si a un modelo muy grande le preguntas sobre un tipo de contenido muy específico, del que no ha visto mucho, no ofrecerá buenos resultados.

1.2 GPT-2

Los LLMs (Modelos de Lenguaje con Aprendizaje Profundo, por sus siglas en inglés) como GPT-2 no producen texto directamente, sino que generan lo que se conoce como "logits" o valores de salida. Para comprender este proceso, es importante entender algunos conceptos clave:

1. **Modelo de lenguaje:** Un modelo de lenguaje es una red neuronal profunda que ha sido entrenada para predecir la probabilidad de que una secuencia de palabras siga a otra en un texto. En el caso de GPT-2, es un modelo de lenguaje basado en la arquitectura de Transformers.

2. **Tokenización:** Antes de alimentar el texto a un LLM, el texto se divide en unidades más pequeñas llamadas "tokens". Los tokens pueden ser palabras completas o fragmentos más pequeños, como subpalabras. Cada token es representado numéricamente para que la red neuronal pueda procesarlo.

3. **Capas de atención:** Los LLMs, como GPT-2, tienen capas de atención que se utilizan para entender la relación entre los tokens en una secuencia. Estas capas permiten al modelo aprender las dependencias y relaciones entre las palabras en el texto.

4. **Logits:** Los logits son los valores de salida de la red neuronal antes de que se aplique una función de activación como la función softmax. Cada logit corresponde a la probabilidad de que un token en particular sea el siguiente en la secuencia dada la secuencia de tokens anteriores.

1.3 Análisis de como los logits se convierten en texto

Para convertir los logits generados por un Modelo de Lenguaje con Aprendizaje Profundo (LLM) en texto legible, se aplica un proceso llamado "decodificación". La decodificación es la etapa final del proceso de generación de texto (que explicaremos mas detalladamente en otro apartado) y es esencial para obtener una secuencia de palabras coherentes y comprensibles a partir de las probabilidades representadas por los logits. Aquí hay un análisis de cómo se realiza este proceso:

1. **Cálculo de probabilidades:** Los logits representan las probabilidades condicionales de que cada posible token (palabra o subpalabra) sea el siguiente en la secuencia, dado el contexto de tokens anteriores. Cada logit se asocia con un token específico y cuantifica cuán probable es que ese token sea el siguiente en la secuencia. Estas probabilidades no están directamente en forma de texto, sino que son valores numéricos.

2. **Función softmax:** Antes de realizar la decodificación, los logits se pasan por una función llamada "softmax". La función softmax toma los valores numéricos de los logits y los convierte en probabilidades. Esto se hace para que las probabilidades sumen 1 y sean más fáciles de interpretar. La fórmula del softmax para un logit z es:

$$P(y = i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

   Donde $P(y = i)$ es la probabilidad de que el token i sea el siguiente en la secuencia, e es la base del logaritmo natural, z_i es el logit para el token i, y n es el número total de tokens posibles.

En resumen, los LLMs como GPT-2 no generan texto directamente, sino que **producen logits como salida**. Estos logits se utilizan para calcular las probabilidades de las palabras siguientes en una secuencia, y a partir de esas probabilidades, se elige la palabra siguiente (la de la probabilidad mas alta, **"argmax"**). Este proceso se repite secuencia por secuencia para generar texto coherente y natural.

2 Proceso de Generación de Texto

1. **Entrada:** Se proporciona una secuencia de tokens iniciales como entrada al modelo. Por ejemplo, si la secuencia de entrada es **"I have a dream"**(ejemplo del documentoa), el modelo calcula las probabilidades de las palabras que podrían seguir a esta secuencia.

2. **Cálculo de logits:** La red neuronal realiza cálculos matemáticos para calcular los logits para cada posible token de siguiente palabra. Estos logits son valores numéricos que indican cuán probable es que cada token siga a la secuencia de entrada.

3. **Muestreo:** Una vez que se obtienen las probabilidades a través de la función softmax, se utiliza algún método de muestreo para seleccionar el token siguiente en la secuencia. Los métodos comunes de muestreo incluyen:

   - **Muestreo aleatorio:** Se elige aleatoriamente un token de acuerdo con las probabilidades obtenidas a través del softmax. Esto introduce variabilidad en la generación y puede dar lugar a textos más diversos.

   - **Muestreo basado en la probabilidad máxima:** Se selecciona el token con la probabilidad más alta. Esto tiende a generar texto más predecible y coherente, ya que siempre se elige la opción más probable.

   Es importante destacar que la elección del método de muestreo influye en el comportamiento y la diversidad del texto generado. Un muestreo más aleatorio tiende a producir textos más creativos pero a veces menos coherentes, mientras que un muestreo basado en la probabilidad máxima tiende a ser más coherente pero potencialmente menos diverso

4. **Retroalimentación:** Después de seleccionar un token, se agrega a la secuencia generada hasta el momento, y el proceso se repite para generar el siguiente token. Este proceso continúa hasta que se alcance una longitud de texto deseada o se cumpla alguna otra condición de finalización.

2.1 Representación ilustrativa de la conversión de token a ID de token y viceversa

Podemos observar el proceso de la conversion de token para tener una cuenta e iniciar sesión en alguna web

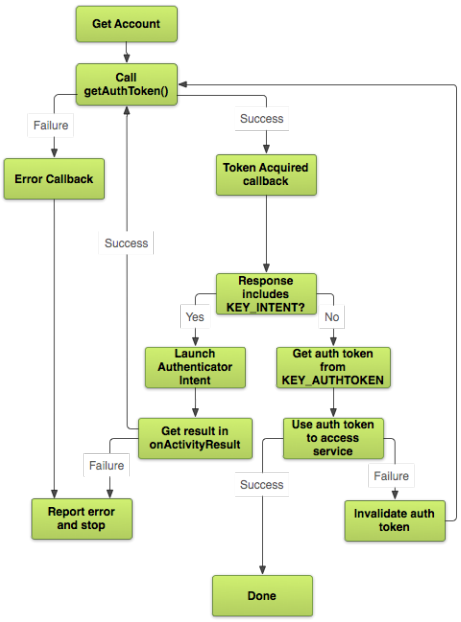

Figure 1: Proceso Tokenización

3 Estrategias de Decodificación

Las estrategias de decodificación se utilizan en el procesamiento de lenguaje natural y en particular en la generación de texto con modelos de lenguaje para decidir cómo convertir las distribuciones de probabilidad de palabras (logits) en secuencias de texto coherentes.

3.1 Descripción y análisis de la búsqueda codiciosa y la búsqueda de haz

Búsqueda Codiciosa (Greedy Search)

Es una estrategia de decodificación en la que, en cada paso de generación, se selecciona la palabra más probable según el modelo de lenguaje y se agrega a la secuencia de salida. En otras palabras, en cada paso, el modelo elige la opción con la probabilidad más alta.

Características y Ventajas:

- Es una estrategia simple y eficiente, ya que solo se selecciona la opción más probable en cada paso.
- Requiere menos tiempo de procesamiento en comparación con la búsqueda de haz.

- Puede funcionar bien en situaciones donde la coherencia y la fluidez del texto son críticas.

Desventajas:

- La búsqueda codiciosa no considera alternativas y no tiene en cuenta la diversidad de las posibles salidas. Esto puede llevar a resultados predecibles y menos creativos.

- Puede generar texto que carece de variedad y originalidad, con una tendencia a la repetición de patrones.

Búsqueda de Haz (Beam Search)
Es una estrategia de decodificación que explora múltiples caminos de generación simultáneamente. En lugar de seleccionar solo la palabra más probable en cada paso, se mantienen las k secuencias más probables (donde k es un hiperparámetro definido por el usuario), y se generan nuevas secuencias a partir de ellas en cada paso.
Características y Ventajas:

- Permite explorar múltiples alternativas y considerar la diversidad en la generación de texto.

- A menudo, produce resultados más diversos y creativos en comparación con la búsqueda codiciosa.

- Puede ayudar a evitar la repetición de patrones y enriquecer la generación de texto.

Desventajas:

- Requiere más tiempo de procesamiento en comparación con la búsqueda codiciosa debido a la exploración de múltiples caminos de generación.

- La elección de k (el tamaño del haz) es un hiperparámetro crítico y debe ser ajustada para equilibrar la coherencia y la diversidad del texto.

En resumen, la búsqueda codiciosa es rápida y simple, pero tiende a generar resultados predecibles y menos creativos. Por otro lado, la búsqueda de haz es más poderosa en términos de explorar diversas alternativas, lo que la hace útil para la generación de texto más diverso y creativo. La elección entre estas dos estrategias depende de los objetivos específicos de generación y las preferencias del usuario.

3.2 Discusión sobre muestreo con top-k y muestreo de núcleo

Muestreo con Top-k (Top-k Sampling)

En el muestreo con top-k, en cada paso de generación, se consideran solo las k palabras con las probabilidades más altas según el modelo. Luego, se elige una palabra entre este subconjunto de k palabras de acuerdo con sus probabilidades. El valor de k se elige como un hiperparámetro y controla la cantidad de opciones consideradas en cada paso.

Características y Ventajas:

- Permite controlar la aleatoriedad al seleccionar solo entre un subconjunto de palabras más probables.
- Ayuda a evitar la selección de palabras extremadamente improbables que pueden llevar a resultados incoherentes o poco naturales.

Desventajas:

- Puede llevar a la generación de texto que tiende a ser predecible y repetitiva, ya que solo se consideran un número limitado de palabras en cada paso.
- El valor de k debe ajustarse cuidadosamente para equilibrar la coherencia y la diversidad del texto.

Muestreo de Núcleo (Nucleus Sampling)

En el muestreo de núcleo, se consideran palabras hasta que la probabilidad acumulada alcance un umbral, denotado como p (también un hiperparámetro). En cada paso, se consideran todas las palabras disponibles y se calcula la probabilidad acumulada de esas palabras en orden descendente de probabilidad. Cuando la probabilidad acumulada alcanza o supera el umbral p, se detiene el proceso de selección.

Características y Ventajas:

- Permite controlar la diversidad y la coherencia del texto, ya que se consideran palabras hasta que se alcance un cierto nivel de probabilidad acumulada.
- Evita la restricción rígida de seleccionar solo un subconjunto de palabras, lo que puede ser útil en la generación de texto más variado.

Desventajas:

- La elección del valor de p es crucial y debe ajustarse cuidadosamente. Un valor muy bajo puede llevar a resultados poco coherentes, mientras que un valor muy alto puede hacer que el texto sea demasiado restringido.

En resumen, el muestreo con top-k es más restrictivo, ya que considera solo un número fijo de palabras más probables. Esto tiende a generar texto más

coherente pero potencialmente menos diverso. El muestreo de núcleo es más flexible, ya que considera palabras hasta que se alcance un umbral de probabilidad acumulada. Esto puede generar texto más diverso, pero es necesario ajustar el umbral para obtener el equilibrio deseado entre coherencia y diversidad.

La elección entre estas estrategias depende de los objetivos específicos de generación de texto y las preferencias del usuario. Ambas estrategias ofrecen un mayor grado de control sobre el proceso de generación en comparación con el muestreo aleatorio simple, lo que permite adaptar la generación de texto a diferentes escenarios y requisitos.

4 Hiperparámetros y su Manipulación

4.1 TEMPERATURA

En el contexto de la decodificación en inteligencia artificial y procesamiento de lenguaje natural, la "temperatura" se refiere a un parámetro que se utiliza en los modelos de lenguaje como GPT (Generative Pre-trained Transformer). Este parámetro controla la aleatoriedad de las predicciones del modelo. Una temperatura alta (por ejemplo, 1.0) hará que las predicciones sean más aleatorias y, a veces, menos coherentes, mientras que una temperatura baja (por ejemplo, 0.2) hará que las predicciones sean más determinísticas y coherentes.

4.2 N-BEAMS

El parámetro num_beams se refiere al número de "rayos" o hipótesis que el modelo considera durante la generación de texto. Básicamente, cuando num_beams es mayor que 1, el modelo considera múltiples posibilidades para la siguiente palabra y luego selecciona la más probable.

Esto puede llevar a un texto de salida más coherente y gramaticalmente correcto, ya que el modelo está considerando más opciones y eligiendo la más probable. Sin embargo, también puede aumentar el tiempo de generación, ya que el modelo está realizando más cálculos.

Ejemplo: si num_beams es igual a 4, el modelo considerará cuatro posibles continuaciones para cada paso de generación y seleccionará la más probable en función de sus ponderaciones de probabilidad. Esto puede ayudar a evitar resultados incoherentes o poco naturales.

4.3 TOP-K

top_k controla el número de las palabras más probables que el modelo considera en cada paso de la generación de texto. Si top_k es igual a 10, por ejemplo, el modelo solo considerará las 10 palabras con las probabilidades más altas para la siguiente palabra en la secuencia.

Este parámetro es útil para limitar la generación a un conjunto más pequeño de opciones, lo que puede hacer que el texto resultante sea más coherente y

significativo. Al restringir las posibilidades a las palabras más probables, se evita la generación de secuencias poco naturales o fuera de contexto.

Es importante mencionar que top_k trabaja en conjunto con temperatura. Mientras que temperatura controla la aleatoriedad de las predicciones, top_k controla cuántas opciones se consideran en cada paso.

Ejemplo: si top_k es 50 y temperatura es 0.8, el modelo considerará las 50 palabras más probables, pero aplicará una cierta dosis de aleatoriedad a esas opciones.

4.4 TOP-P

top_p, también conocido como "núcleo de probabilidad", es otro parámetro utilizado en estrategias de decodificación en procesamiento de lenguaje natural.

Este parámetro se utiliza para controlar la cantidad acumulada de probabilidad que se considera durante la generación de texto. Si top_p está configurado en 0.9, por ejemplo, el modelo considerará las palabras cuya probabilidad acumulada supere el 90/100 del total.

En otras palabras, top_p permite mantener un conjunto dinámico de opciones, adaptando el número de posibilidades a la distribución de probabilidad de las palabras en un determinado contexto. Esto es especialmente útil en situaciones en las que la probabilidad de las palabras varía significativamente.

top_p se utiliza en conjunto con la temperatura y el top_k para ajustar el proceso de generación de texto.

Ejemplo: si top_p es bajo y temperatura es alto, el modelo será más creativo pero aún así estará restringido a un conjunto relativamente pequeño de palabras probables.

4.5 CONCLUSIONES

La temperatura en el contexto de estrategias de decodificación controla la aleatoriedad en las predicciones del modelo.

num_beams determina cuántas hipótesis considera el modelo en cada paso de la generación de texto.

top_k limita el conjunto de palabras consideradas a las k con las probabilidades más altas, evitando así la generación de secuencias poco naturales o fuera de contexto.

Por su parte, top_p controla la cantidad acumulada de probabilidad que se considera durante la generación de texto, permitiendo ajustar el número de opciones en función de la distribución de probabilidad de las palabras en el contexto actual.

Estos parámetros, cuando se utilizan en conjunto y ajustados correctamente, permiten afinar el comportamiento de los modelos de lenguaje para obtener resultados deseados, equilibrando la creatividad, coherencia y relevancia en las salidas generadas. Cada uno desempeña un papel crucial en la configuración del proceso de decodificación.