TOC-6 (ejercicios 1,2 y 3)

Carlos Puigserver y Germán Llorente

October 2023

1 Descripción Detallada de LLMs

Los Modelos de Lenguaje Basados en Aprendizaje Profundo (LLMs) como GPT-2 son una clase de modelos de procesamiento de lenguaje natural que se han vuelto muy populares en diversas aplicaciones, como la generación de texto, la traducción automática y la respuesta a preguntas. Para comprender cómo funcionan, es esencial comprender el concepto de "logits" y cómo se traducen en texto legible.

1.1 Logits en LLMs

Los LLMs, incluido GPT-2, no producen texto directamente. En cambio, generan lo que se conoce como "logits". Los logits son valores numéricos que representan la probabilidad de ocurrencia de cada palabra o token en un vocabulario dado en un contexto específico. En otras palabras, son una representación numérica de la confianza del modelo en las diferentes palabras que podrían seguir en una secuencia de texto.

Los logits son fundamentales en la generación de texto, ya que reflejan la incertidumbre del modelo al predecir las palabras siguientes. Cuanto mayor sea el valor del logit para una palabra, mayor será la probabilidad de que esa palabra sea la siguiente en la secuencia.

1.2 Traducción de Logits a Texto

La traducción de logits a texto legible es una etapa crucial en la generación de texto por parte de los LLMs. Este proceso generalmente implica el uso de una función llamada "softmax". La función softmax toma los logits como entrada y los convierte en una distribución de probabilidad sobre el vocabulario. Esto significa que asigna probabilidades a todas las palabras posibles en función de los logits.

La función softmax es una función matemática que suaviza los logits y los convierte en probabilidades normalizadas. Esta transformación es necesaria para convertir las salidas del modelo en una distribución de probabilidad comprensible.

Matemáticamente, dada una secuencia de logits $L = [l_1, l_2, \ldots, l_n]$ correspondientes a un conjunto de palabras candidatas, la función softmax se calcula de la siguiente manera:

$$P(w_i|L) = \frac{e^{l_i}}{\sum_{j=1}^{n} e^{l_j}} \tag{1}$$

Donde: - $P(w_i|L)$ es la probabilidad de que la palabra w_i sea la siguiente en la secuencia. - e es la función exponencial. - $\sum_{j=1}^{n} e^{l_j}$ es la suma de las exponenciales de todos los logits.

Una vez que se ha calculado la distribución de probabilidad sobre el vocabulario, se selecciona la palabra con la probabilidad más alta como la siguiente palabra en la secuencia generada. Este proceso se repite para generar texto secuencialmente.

La elección de la palabra con la probabilidad más alta se conoce como "argmax" y es una estrategia común para la generación de texto en LLMs.

En resumen, los LLMs como GPT-2 generan logits como una representación numérica de la probabilidad de las palabras en un contexto dado. Estos logits se traducen en texto utilizando la función softmax para generar una distribución de probabilidad sobre el vocabulario y seleccionar las palabras con mayor probabilidad. Esto permite que los LLMs generen texto coherente y contextualmente relevante.

2 Proceso de Generación de Texto

El proceso de generación de texto en modelos de lenguaje basados en aprendizaje profundo, como GPT-2, es un procedimiento complejo que abarca desde la entrada de texto hasta la predicción del siguiente token. A continuación, se describe detalladamente este proceso y se presenta una representación ilustrativa de la conversión de tokens a IDs de tokens y viceversa.

2.1 Descripción del Proceso

2.1.1 Entrada de Texto

El proceso comienza con la entrada de texto proporcionada al modelo. Esta entrada puede ser una oración o un fragmento de texto en lenguaje natural que se utiliza como contexto inicial para la generación de texto. El modelo utiliza este contexto para generar una secuencia de texto continua.

El contexto inicial es esencial para que el modelo comprenda el tema y el estilo del texto que debe generar. Puede influir en gran medida en el contenido generado.

2.1.2 Tokenización

El texto de entrada se somete a un proceso de tokenización. La tokenización implica dividir el texto en unidades más pequeñas llamadas "tokens". Los tokens

pueden ser palabras individuales, subpalabras o caracteres, dependiendo de la configuración del modelo. Cada token se representa mediante un identificador único conocido como "ID de token". Esto facilita el manejo de texto en formato numérico.

La tokenización es un paso esencial para que el modelo trabaje con texto, ya que convierte el texto en una secuencia estructurada de unidades que el modelo puede procesar.

2.1.3 Generación de Texto

Una vez que el texto se ha tokenizado y convertido en una secuencia de IDs de tokens, el modelo comienza a generar texto adicional. El proceso de generación implica predecir el siguiente token en función de los tokens previos en la secuencia. El modelo utiliza una función de puntuación (logits) para evaluar la probabilidad de cada token posible como el siguiente en la secuencia.

La elección del siguiente token es crucial y se basa en la probabilidad estimada por el modelo. Puede influir en la coherencia y el significado del texto generado.

2.1.4 Muestreo de Tokens

Para determinar el siguiente token, el modelo puede utilizar diferentes estrategias de muestreo, como el muestreo de softmax, la maximización de la probabilidad o la búsqueda de haz. Estas estrategias influyen en la diversidad y coherencia del texto generado.

El tipo de estrategia de muestreo seleccionada puede afectar el estilo de generación del modelo. El muestreo de softmax, por ejemplo, puede introducir variabilidad en la elección de palabras.

2.1.5 Generación Continua

El proceso de generación de texto continúa de manera iterativa, donde el token predicho se agrega a la secuencia y se utiliza como contexto para predecir el siguiente token. Este proceso se repite hasta que se genera la longitud deseada de texto o se alcanza un token de finalización especial.

La generación continua permite que el modelo cree texto coherente y completo, que puede ser de cualquier longitud especificada por el usuario.

2.2 Representación Ilustrativa

La conversión de tokens a IDs de tokens y viceversa se ilustra a continuación:

La representación ilustrativa muestra cómo el modelo convierte tokens, como palabras o subpalabras, en identificadores numéricos únicos. Esta conversión es esencial para que el modelo procese el texto de manera eficaz.

En resumen, el proceso de generación de texto en modelos de lenguaje basados en aprendizaje profundo implica la entrada de texto, tokenización, generación iterativa de texto y la conversión entre tokens y sus correspondientes

IDs de tokens. Esta representación ilustrativa proporciona una visión general de cómo funciona este proceso.

3 Estrategias de Decodificación

Los modelos de lenguaje, como GPT-2, son capaces de generar texto coherente y relevante a partir de un contexto dado. Sin embargo, para lograr esto, es necesario utilizar estrategias de decodificación adecuadas. A continuación, exploraremos dos de las estrategias más comunes: la **búsqueda codiciosa** y la **búsqueda de haz**, así como las técnicas de **muestreo con top-k** y **muestreo de núcleo**.

3.1 Búsqueda Codiciosa (Greedy Decoding)

La búsqueda codiciosa es una estrategia de decodificación simple pero efectiva en la que, en cada paso, se elige la palabra o token con la mayor probabilidad de ocurrencia según el modelo. En otras palabras, en cada paso, se selecciona la opción más probable sin considerar las opciones futuras. Esta estrategia tiende a generar texto coherente, pero puede carecer de diversidad y originalidad, ya que siempre elige la opción más segura.

Matemáticamente, dado un conjunto de tokens candidatos C_t en el paso t, la búsqueda codiciosa selecciona:

$$w_t = \arg\max_{w_i \in C_t} P(w_i|\text{contexto}) \tag{2}$$

Donde: - w_t es el token seleccionado en el paso t. - $P(w_i|\text{contexto})$ es la probabilidad condicional del token w_i dada la secuencia de contexto.

La ventaja de la búsqueda codiciosa es su eficiencia computacional, ya que no requiere explorar múltiples opciones. Sin embargo, puede generar resultados

predecibles y repetitivos.

3.2 Búsqueda de Haz (Beam Search)

La búsqueda de haz es una estrategia más avanzada que busca mejorar la diversidad y calidad del texto generado. En lugar de seleccionar la opción más probable en cada paso, la búsqueda de haz mantiene un conjunto de k hipótesis o secuencias parciales en cada paso, donde k es un hiperparámetro llamado "ancho del haz". En cada paso, se exploran las k opciones más probables y se mantienen las mejores k secuencias parciales.

Matemáticamente, en el paso t, la búsqueda de haz selecciona el conjunto B_t de las k mejores secuencias parciales basadas en sus probabilidades conjuntas:

$$B_t = \text{top-k}\left(\bigcup_{i=1}^{k}\{(\text{secuencia}_i, P(\text{secuencia}_i|\text{contexto}))\}\right) \tag{3}$$

Donde: - top-k(S) selecciona las k mejores opciones de un conjunto S en función de las probabilidades conjuntas. - secuencia$_i$ es una secuencia parcial. - P(secuencia$_i$|contexto) es la probabilidad conjunta de la secuencia parcial.

La búsqueda de haz permite explorar múltiples posibilidades y puede generar resultados más variados y contextuales. Sin embargo, aumenta la complejidad computacional debido a la gestión de múltiples hipótesis.

3.3 Muestreo con Top-k y Muestreo de Núcleo

El muestreo con top-k y el muestreo de núcleo son estrategias que buscan equilibrar la exploración y la calidad en la generación de texto. Estas estrategias introducen aleatoriedad al seleccionar el siguiente token en lugar de elegir siempre la opción más probable.

- El **muestreo con top-k** selecciona aleatoriamente el siguiente token de entre los k tokens con las mayores probabilidades. Esto permite cierta diversidad y evita la previsibilidad de la búsqueda codiciosa.

- El **muestreo de núcleo** (nucleus sampling) es similar, pero en lugar de seleccionar siempre los k tokens principales, selecciona tokens hasta que su probabilidad acumulada alcance un cierto umbral, definido como una fracción α de la probabilidad total.

Estas estrategias permiten la generación de texto más variado y pueden ser útiles cuando se busca originalidad en lugar de coherencia absoluta. Sin embargo, también pueden generar resultados menos coherentes en algunos casos.

En resumen, las estrategias de decodificación, como la búsqueda codiciosa, la búsqueda de haz, el muestreo con top-k y el muestreo de núcleo, juegan un papel crucial en la generación de texto en modelos de lenguaje. La elección de la estrategia adecuada depende de los objetivos específicos de generación, equilibrando coherencia y diversidad.