Tarea 4 - Reporte de modelo generativo

Residual Energy-Based Models for Text Generation

Camilo Carvajal Reyes

16 de diciembre de 2022

Residual Energy-Based Models for Text Generation, por Deng et al. 2019 [1], es un trabajo en el cual se propone un método para mejorar la generación de texto que viene de un modelo de lenguaje pre-entrenado (que llamaremos ML). La idea general es entrenar un discriminador que detecte si una frase es natural o bien generada con un ML. Con esto, la distribución propuesta por el ML será la *proposal distribution* y el discriminador ponderará las probabilidades para luego re-samplear como en *importance sampling*. Formalmente el modelo generativo a considerar es, para una secuencia de tokens x:

$$p_\theta(x) = p_{ML}(x) \cdot \frac{\exp(-E_\theta(x))}{\int p_{ML}(z)\exp(-E_\theta(z))dz} \propto p_{ML}(x) \cdot \exp(-E_\theta(x)) \,, \tag{1}$$

con lo cual el entrenamiento se reduce a encontrar los parámetros θ de la energía que asignen baja energía a texto humano. Esto se logra de manera eficiente usando *conditional noise contrastive estimation* (conditional NCE). Finalmente, la formulación residual hace posible la estimación de la perplejidad mediante cotas del factor normalizador, las cuales muestran una mejora empírica de esta métrica de calidad de lenguaje generado con respecto a MLs clásicos.

1. Modelos Energy-Based

Recordemos el principio de los modelos basados en energía vistos en clase. Queremos aproximar una distribución desconocida. Para esto consideramos una función E_θ a determinar y que llamaremos energía (en el curso esta se denotó f_θ). Luego, aprovechando las propiedades de la exponencial, obtenemos una distribución de probabilidad en la siguiente expresión:

$$p_\theta(x) = \frac{\exp(-E_\theta(x))}{\int \exp(-E_\theta(z))dz} \,.$$

Normalmente el denominador es intratable, sin embargo en muchos contextos no nos importará su valor real. Un ejemplo que será relevante en este contexto es el siguiente: sea un conjunto $\{x_1, \ldots, x_n\}$, queremos saber la probabilidad de x_j relativa al resto de los miembros del conjunto con respecto a la medida p_θ. Un poco de álgebra hará que el término de la integral se cancele, con lo que la expresión queda fácilmente computable:

$$\frac{p_\theta(x_j)}{\sum_{i=1}^n p_\theta(x_i)} = \frac{\exp(-E_\theta(x_j))}{\sum_{i=1}^n \exp(-E(_\theta(x_i)))} \tag{2}$$

Esto hace que estos modelos sean particularmente útiles para detección de ruido o anomalías. Por lo demás, el entrenamiento de estos modelos se lleva a cabo maximizando la máxima verosimilitud empírica, que a su vez equivale a minimizar la divergencia KL entre nuestra distribución parametrizada y la distribución real de datos. Antes de adentrarnos en como aplicar estos principios junto a un modelo de lenguaje introduciremos ciertos conceptos esenciales para el modelamiento de texto.

2. Modelos generativos para texto

Formalmente, un modelo de lenguaje (ML) corresponde a una distribución de probabilidad sobre secuencias de palabras (o tokens) pertenecientes a un cierto vocabulario V. *GPT* (Radford et al. 2019 [2]) pertenece a un tipo de modelos que han tenido rotundo éxito en los últimos años. Estos están basados en la arquitectura neuronal *transformers* (Vaswani et al. 2017 [3]). *Transformers* es un ejemplo de modelo autoregresivo profundo, i.e., modelos generativos secuenciales que predicen valores futuros de una sucesión dado su pasado. Más precisamente, los modelos que nos conciernen son aquellos que modelan las probabilidades de la secuencia de manera siguiente:

$$p_{ML}(x_{p+1},\dots,x_T|x_1,\dots,x_p) = \prod_{i=p+1}^{T} p(x_i|x_1,\dots,x_{i-1}) = \prod_{i=p+1}^{T} \frac{s(x_1,\dots,x_p,\dots,x_i)}{\sum_{y\in V} s(x_1,\dots,x_p,\dots,x_{i-1},y)}\,, \tag{3}$$

donde $s(x_1,\dots,x_p,\dots,x_{i-1},y)$ es un el score del token y dado el input $x_1,\dots,x_p$ y el historial de predicción $x_{p+1},\dots,x_{i-1}$. A esto se le llama "normalización local". De esta manera, los modelos de lenguaje son capaces de producir lenguaje de manera secuencial (token por token), además de ser útiles para otras tareas usando transfer-learning. Los MLs se entrenan en grandes corpuses de texto al maximizar la verosimilitud. En lo que sigue asumiremos un ML que obedece este setting general sin entrar en más detalles.

3. Modelo de energía residual

Pese a su éxito en generación de lenguaje, estos modelos no son perfectos. Como los tokens deben ser generados de a uno, se está propenso a la propagación de errores. A esto se le llama "*exposure bias*", el cual fue introducido por Ranzato et al. (2016) en el contexto de generación con redes recurrentes [4]. Es por esto que se considerará un modelo residual para mejorar la distribución de probabilidad inducida por un ML. Se le llama residuo a la diferencia entre un valor real y su predicción. Acá se tratará de modelar la diferencia entre la distribución real del lenguaje natural a aquella de un ML utilizando un discriminador. La idea del modelo de la ecuación 1 está entonces basada en el trabajo de Grover et al. (2019) [5], que propone la utilización de un discriminador entre datos artificiales y reales para así corregir el sesgo de un modelo generativo en un contexto cualquiera. El modelo resultante es el siguiente (que es el modelo 1 pero usando la nomenclatura secuencial condicional):

$$p_\theta(x_{p+1},\dots,x_T|x_1,\dots,x_p) = \frac{p_{ML}(x_{p+1},\dots,x_T|x_1,\dots,x_p)\exp(-E_\theta(x_1,\dots,x_T))}{Z_\theta(x_1,\dots,x_p)}\,, \tag{4}$$

donde $Z_\theta(x_1,\dots,x_p)$ es un parámetro normalizador que integra las continuaciones de secuencias posibles dada la secuencia parcial $x_1,\dots,x_p$. Este valor, que es intratable, no será siempre de nuestro interés, como se planteó en la sección 1, sólo lo estimaremos cuando realicemos la evaluación. Nuestro objetivo entonces se reduce a encontrar una buena función de energía E_θ que discrimine entre texto real y generado por el ML. De este modo, se le asignará más energía a aquellas secuencias que difieran más de la distribución real y nos acercaremos entonces a la distribución real del lenguaje. Esta energía será parametrizada e inicializada del mismo modo que un modelo de lenguaje, en particular usando la arquitectura *transformers* [3].

4. Entrenamiento

Noise Contrastive Estimation (NCE), que fue introducido por Gutmann et Hyvärinen (2010) [6], es la técnica usada para el entrenamiento de la energía, en su versión condicional Es un método especialmente práctico para estimación con modelos no-normalizados, como lo son los modelos basados en energía. Esto puesto que la estimación de máxima verósimilitud requiere sampleos de MCMC que son muy costosos. El método cNCE entrena un clasificador binario sobre las probabilidades de dos modelos: uno el modelo a

optimizar, que en este caso es el de la ecuación 4, y por otro lado una distribución de "ruido", que será nuestro ML de base. Notemos por un lado que $\log p_\theta - \log p_{ML} = -E_\theta$. Tenemos la función a maximizar es la siguiente:

$$\max \mathbb{E}_{x_+ \sim P_{data}} \log \left(\frac{1}{1 + \exp(E_\theta(x_+))} \right) + E_{x_- \sim p_{LM}} \log \left(\frac{1}{1 + \exp(-E\theta(x_-))} \right), \tag{5}$$

donde x_+ es texto natural de nuestros datos de entrenamiento, y x_- es una secuencia sampleada del ML (dada una secuencia de prefijo). Tenemos garantías teóricas de que el máximo de la ecuación 5 se alcanza en $\log p_{ML} - E_\theta = \log p_{data}$ si es que los soportes de p_{data} y p_{LM} coinciden y si es que existe tal θ [6]. Si pudiésemos garantizar esta existencia entonces tendríamos igualdad en la ecuación 1, i.e., $p_\theta(x) = p_{ML}(x) \cdot \exp(-E_\theta(x))$, pero en la practica la normalización debe ser estimada.

5. Generación

Una opción para samplear de la distribución aprendida es usar el approach típico de modelos de normalización-local (i.e., aquellos donde se obedecen las probabilidades condicionales de la igualdad de la izquierda en la ecuación 3) sería calcular las probabilidades paso a paso:

$$p(x_t|x_1, \ldots, x_{t-1}) = p_{LM}(x_t|x_1, \ldots, x_{t-1}) \frac{\mathbb{E}_{x'_{t+1}, \ldots, x'_T \sim P_{LM}(\cdot|x_1, \ldots, x_t)}[\exp(-E_\theta(x_1, \ldots, x_t, x'_{t+1}, \ldots, x'_T))]}{\mathbb{E}_{x'_t, \ldots, x'_T \sim P_{LM}(\cdot|x_1, \ldots, x_{t-1})}[\exp(-E_\theta(x_1, \ldots, x_{t-1}, x'_t, \ldots, x'_T))]} \tag{6}$$

Esto correspondería a una manera auto-regresiva de generar las secuencias, pero es ineficiente. En la práctica se usa el método de *importance sampling auto-normalizado* de Grover et al. (2019) [5]. Acá sampleamos de nuestra proposal distribution (nuestro ML) y resampleamos de acuerdo a la función de energía. Esto se muestra en el siguiente algoritmo, que se denomina *top-k joint sampling*. Nótese que el procedimiento utiliza lo planteado en ecuación 2.

Código 1: Algoritmo *top-k Joint Sampling*

Input: número n de sampleos, valor de k

Obtener sampleos $\{x^1, \ldots, x^n\}$ de P_{ML} // con top k sampling

Calcular energías $s^i = E_\theta(x^i)$ para cada $x^i \in \{x^1, \ldots, x^n\}$

Re-samplear x desde $\{x^1, \ldots, x^n\}$, donde un sample x^i tiene probabilidad $x^i = \frac{\exp(-s^i)}{\sum_{j=1}^n \exp(-s^j)}$

return x

---

6. Evaluación

La métrica de evaluación para la generación de texto que se suele usar es la *perplejidad* (PPL), que se define como $2^{-\frac{1}{T-p} \sum_{i=p+1}^{T} \log_2 p(x_i|x_{i-1}, \ldots, x_1)}$. Su interpretación es el promedio de tokens de los cuales el modelo está inseguro en cada paso de generación. Esto es equivalente a elevar dos por la entropía cruzada entre la distribución real y la del modelo. La log-verosimilitud de la expresión implica estimar el factor normalizador. Por ende en el artículo se propone las siguientes cotas:

Proposición
Sea $T_n = \log \frac{1}{n} \sum_{i=1}^n \exp(-E(x_i))$ *el estimador empírico de* $\log(\mathbb{E}_{x \sim p_{LM}} \exp(-E(x)))$, *donde* $\{x_1, \ldots, x_n\}$ *son sampleos del modelo de lenguaje. Entonces*

$$Z_\theta - \epsilon < \mathbb{E}[T_n] < Z_\theta < \mathbb{E}[(2n-1)T_n - 2(n-1)T_{n-1}] < Z_\theta + \epsilon \tag{7}$$

7. Resultados y conclusiones

Como primer análisis, en figura 1 se vislumbra que los histogramas de sampleos de del modelo conjunto calza más con la distribución real de los datos comparado con el ML.

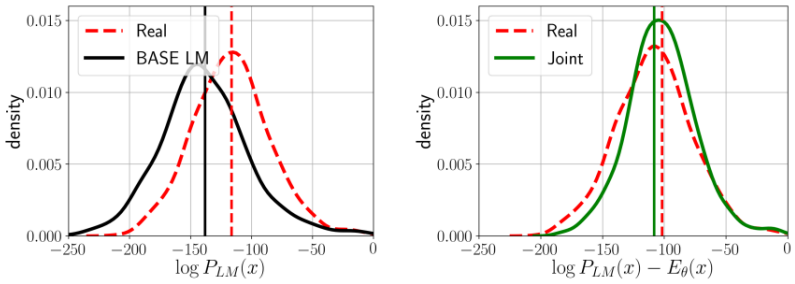

Figura 1: Densidad de log-probabilidades.

Por lo demás, se puede computar un rango para la perplejidad utilizando las cotas de la ecuación 7 (en este caso utilizando 100000 sampleos). Estas muestran un mejor resultado que los distintos modelos de lenguaje. Esto se mantiene además cuando se comparan MLs que tienen aproximadamente la misma cantidad de parámetros que los modelos conjunto (un ML más la red neuronal de energía). Más aún, cuando las oraciones generadas por los distintos modelos se someten a evaluación humana, estos muestran mejoras, aunque no rotundas (preferencia del modelo conjunto alrededor de 55 % de las veces).

Es importante recalcar las limitaciones que posee este tipo de modelos. Primeramente, como se usa el modelo de lenguaje de base como *proposal distribution*, si los sampleos de este son malos, la energía podrá empujar la elección de los "menos malos", pero malos al fin y al cabo. Este método si bien induce mejoras, depende de una buena calidad de ML, lo cual puede ser una hipótesis fuerte (pese al éxito que estos han tenido). Dicho esto, el modelo es un interesante método de generación de texto y que utiliza MLs pre-entrenados de manera eficiente.

8. Referencias

[1] Deng, Y., Bakhtin, A., Ott, M., Szlam, A., Ranzato, M. (2019, September). Residual Energy-Based Models for Text Generation. International Conference on Learning Representations. https://openreview.net/forum?id=B1l4SgHKDH

[2] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I. (2019, February 24). Language Models are Unsupervised Multitask Learners. OpenAI Blog. https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf

[3] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., Polosukhin, I. (2017). Attention is All you Need. Advances in Neural Information Processing Systems, 30, 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[4] Ranzato, M. A., Chopra, S., Auli, M., Zaremba, W. (2016). Sequence level training with recurrent neural networks. In 4th International Conference on Learning Representations, ICLR 2016. https://arxiv.org/abs/1511.06732

[5] Grover, A., Song, J., Kapoor, A., Tran, K., Agarwal, A., Horvitz, E. J., Ermon, S. (2019). Bias Correction of Learned Generative Models using Likelihood-Free Importance Weighting. Advances in Neural Information Processing Systems, 32. https://proceedings.neurips.cc/paper/2019/hash/d76d8deea9c19cc9aaf2237d2bf2f785-Abstract.html

[6] Gutmann, M., Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 297–304. https://proceedings.mlr.press/v9/gutmann10a.html