NOTE ON ENTROPIC LLMS -I

VAIBHAV KALVAKOTA

Abstract. A Note on entropy in LLMs. Maybe I will write more with better implementations.

Entropy is a very fascinating term, and people use it in many different settings. I worked with generalized entropy, you could work with gravitational entropy, someone else could work with some pure information theoretic Shannon entropy, or von Neumann entropy, etc. In LLMs, entropy is an interesting term that definitely has a lot of importance, and Entropix is a high level implementation of entropy based sampling. The purpose of this Note is to provide a very short description, mostly me just fantasizing, what could be the directions for entropy in LLMs.

Most LLMs to date rely on the Transformer neural network architecture introduced by Vaswani et al (2017), which works far better than RNNs and LSTMs in terms of long-term context dependence and generative correlations due to the *self-attention* model, although at an $O(n^2)$ complexity. A simple way of using these models is to use *greedy sampling*, where we simply pick the token with the highest probability. This creates a lot of issues, since the model is very deterministic and could lead to very degenerate outputs. The alternative is to take sampling parameters such as top-k, top-p, min-p and temperature, which allow more flexibility on the sampling strategy.

In order to make sense of dynamic sampling, one could fairly easily guess that somehow the objective is to find the entropy of the model and make sampling "better and better" on probability distributions. That is, we want to take the Shannon entropy, find out if it is higher or lower than some dynamic threshold (which could also be fixed in principle), and then sample the parameters better and better. Entropix [1], currently a Llama 3.1 based model is doing a really interesting project with using entropy/varentropy based sampling methods for LLMs. The very superficial idea is that for some probability for the t^{th} token given a context of $t-1$ length, we calculate the entropy

$$S = -p_t \ \log \ p_t \ ,$$

to optimize sampling strategies for top-p, top-k and the temperature T.

To make all of this more formal, we will define a large language model as the collection $\mathcal{M}(\mathbf{x}, \mathbf{y}, \mathcal{P}_{\text{true}}, \mathcal{Q}, \mathbf{\Delta}, \mathbf{J}, \mathbf{\Theta}, \mathbf{S})$. Here, $\mathbf{x}$ are the inputs, $\mathbf{y}$ are the model outputs, $\mathcal{P}_{\text{true}}$ is the true probability distribution, $\mathcal{Q}$ is the model probability distribution, $\mathbf{\Delta}(\mathcal{P}_{\text{true}}||\mathcal{Q})$ is the corresponding Kullback-Leiblar divergence, $\mathbf{J}$ is the loss function for the model, $\mathbf{\Theta}$ are the parameters of the model and $\mathbf{S}$ are the sampling parameters $(T, p_{\text{min}}, k_{\text{top}} \dots)$. This is to say that we are working with an energy based model (EBM) for which we want to calculate the energy function

Date: November 8, 2024.

and minimize it, and update $\mathbf{S}$ dynamically to get better and better generative predictions [2]. In our case, the energy function E is just the usual cross entropy loss function $\mathbf{J}_{CE}$, and we want to minimize the KL-divergence so that the entropy sampling becomes redundant (optimal model outputs). Alternatively, you could view the objective here to be to minimize the *free energy* instead, which has the same effect, only that it encapsulates more sampling redundancy as well. Loosely, start by noticing that $p(x) = \frac{\exp(-\beta E)}{Z}$, and the entropy

$$S(p) = -\int p(x) \log p(x) \ . \tag{1}$$

Expand into

$$S(p) = \beta^{-1} \mathbb{E}_p[E(x)] + \mathcal{F}_\beta \ , \tag{2}$$

where we just substituted for $p(x)$ and identified the Helmholtz free energy as the log of the partition function Z_β. Then, the sampling algorithms are solved in terms of $\min E$: start by computing $E(\mathcal{T}_i)$, and then find min-p as usual. However, you could instead just seek to minimize the free energy, since it implies selecting higher probability choices and tells you what the entropy is anyway. Working with $\mathcal{F}_\beta$ rather than just E is in general better because it incorporates β and gives a more subtle relationship between the parameters and the entropy.

There are also possibilities with adaptive sparsity in Transformers based on entropy evaluations [3]. You start by taking the entropy and define something like

$$\zeta_\alpha(S) = \begin{cases} \zeta_{\min} \equiv 0 & \text{High entropy} = \text{softmax} \\ \mathcal{D}(S) & \mathcal{S}_0 \leq S \leq \mathcal{S}_{\max} \\ \zeta_{\max} & \text{Low entropy} = \text{sparsemax} \end{cases} \ , \tag{3}$$

where $\mathcal{D}(S)$ is a function of entropy that ranges from $\zeta_{\min} = 0$ to $\zeta_{\max}$. This is just being bound between the usual softmax and sparsemax functions, so basically we are saying that if we have high entropy, we want lower sparsity and if we have lower entropy, we want higher sparsity. This is essentially to employ a version of magnitude-clipping or pruning but the more stronger claim I want to make is that doing this entropy-dynamic sparsity increases the effective correlation of the model, by reducing the correlations between irrelevant and relevant inputs by fixing higher entropy configurations. In most models this is not exactly an issue, but arguably in the efficiency vs performance trade-off, maximizing effective correlation is possible only if we don't restrict ourselves to dense attention weights, meaning that softmax is not very efficient in this sense.

While I have not come up with a very formal definition of $\mathcal{D}(S)$ and how exactly this model gets implemented, here is a reason why this whole entropy-based sparsity does not have to be *that* adaptive, and here is where varentropy comes in, deciding whether or not $\zeta(S)$ has to be a *continuously adaptive parameter*. In calculating the metric attentions, we find entropy as well as varentropy. This decides whether the sparsity factor is dynamic or fixed; low varentropy bounds are exactly softmax and sparsemax. Higher varentropy implies that the bounds to $\mathcal{D}(S)$ changes, although still bounded by the softmax and sparsemax functions. I am still working on how exactly to see this relation and give a mathematical formulation for it.

The next thing I will mention here is the order of complexity for the Transformer model with adaptive entropy sparsity. The usual vanilla Transformer model has an order of complexity that is quadratic in the context length, $O(n^2)$, whereas for a sparsity model, we can obtain a much better complexity order of $O(n\sqrt{n})$. I am not exactly sure if the fixed-ness of the model affects this complexity better or worse in terms of efficiency but on an all, the complexity (at least theoretically) is not very fixed in terms of sparsity and therefore depends on $\zeta_\alpha(S)$:

$$O(\text{complexity}) \propto \frac{1}{\zeta_\alpha(S)} \ . \tag{4}$$

When sparsity is higher, the complexity typically decreases since it has to tend to fewer tokens in the token pool $\mathcal{T}_i$. This depends on the overall varentropy metric calculations, and so we arrive at the following theoretical complexity for the AES:

$$O(\text{complexity}) \propto \frac{1}{\text{varentropy}} \ . \tag{5}$$

It would be nice to actually run a model with AES and see how it changes the complexity and the validations. For now, we still are left with the question of what $\mathcal{D}(S)$ exactly is. Remember though that this function is a continuous function taking values between the softmax with $\zeta(S) = 0$ and sparsemax when $\zeta(S) = \zeta_{\text{max}}$. This definition bounds the entropy sparsity parameter between either extremely dense or extremely sparse attention weights. I used this definition to agree with the α -entmax function, where $\alpha = 1$ is softmax and $\alpha = 2$ is the sparsemax function. This bounding may be changed if appropriate, since the α in our context is a similar learnable parameter of the model that is optimized with stochastic gradient descent.

Finally I think my point is that viewing Entropix as an EBM is very helpful and points to many interesting directions. That's pretty much all I have to say for now.

References

[1] @_xjdr, *Entropix*, *Github* (https://github.com/xjdr-alt/entropix) .
[2] V. Kalvakota, *Boltzmann and Witten*, *Notion* (https://quantum-vaibhav.notion.site/Boltzmann-and-Witten-131b2c48eabf80679bc6d7558b1b0038?pvs=4) .
[3] V. Kalvakota, *Adaptive Entropy Sparsity -I*, *Notion* (https://quantum-vaibhav.notion.site/Adaptive-Entropy-Sparsity-I-12bb2c48eabf80b7bb2ec1b8cb9be68d?pvs=4) .