W. ROSS MORROW

## 1. Introduction

Consider a corpus $\mathcal{C}$ of tokens from a language $\mathcal{T} = \{1, \ldots, T\}$ (w.l.o.g.) that at the very least "covers" $\mathcal{T}$ in the sense that every $t \in \mathcal{T}$ appears in $\mathcal{C}$. By a corpus we basically mean, here, a giant body of text or a "long" sequence of elements of $\mathcal{T}$. Can we compute the empirical distribution of next tokens for some batch size $B$? Let's define what we mean clearly: we want to estimate the conditional probabilities

$$\rho_B(\mathbf{t}, \tau) = \mathbb{P}(\ t_{B+1} = \tau \mid t_1, \ldots, t_B\ )$$

with "(Conditional) Empiricial Frequencies" or (C)EFs.

Before detailing how, why would we do such a thing? A plausible "hypothesis" is that these probabilities are what LLMs like GPT are fundamentally modeling. Any functional representation, such as a variant of multiclass logit regression or the deep networks in modern LLMs, impose structure on sequential predictions. Yet it is possible that (a) for "small" data volumes such structure exists to "smooth" gaps in knowledge of some "true" conditional probabilities, while (b) as the training corpus size grows the model (if from a suitably "universal" class) will increasingly tend to the empirical conditional probabilities and effectively "regurgitate" patterns from its training data. Formally of course there are some statistical assumptions (consistency, unbiasedness) hiding here that we'll ignore.

This idea – that the best a LLM could do is defined by the (C)EFs – has implications for conceptualizing what LLMs are. First, should this hypothesis be correct, it is a formal, almost philosophical illustration of how it is impossible for LLMs to be generatively "innovative". In a strict sense they can only repeat the patterns of the past, as described by the (C)EFs in the training data. Which is not to say they are not still *useful*, by any means; though we should still be able to reflect on the limitations of such tools. Second, if complex structure aims at universality, and there is enough data to faithfully represent the (C)EFs, then the complex structure has to *compress* the (C)EFs. In slightly plainer words: as we increase the volume of data used to train a LLM from a reasonably "universal" class, either training of the LLM parameters must be vastly simpler than computing the (C)EFs directly or sampling sequences from the LLM must be vastly simpler than from the (C)EFs (or both) for the LLM to be "effective". Something like GPT-3 is hardly "parsimonious" being based on over 175 *billion* parameters, with larger models in the works (if not already extant), and generative sampling requiring complex distributed computing on specialized hardware.

At least to me, it is not entirely clear that (C)EFs are "efficiently" computable (in, so to speak, "training"), or sampling from them is efficient, or, most importantly, if samples from (C)EFs appear to mimic "real" language structure like LLMs do (as defined in the training corpus). Here we'll focus on those questions. What datastructure models (C)EFs? How would we (efficiently, scaleably) compute them from a corpus? How would we generate sequences?

## 2. Modeling

Again we want to estimate the (C)EFs of

$$\rho_B(\mathbf{t}, \tau) = \mathbb{P}(\ t_{B+1} = \tau \mid t_1, \ldots, t_B\ )$$

This is probably to be done via something like

$$\hat{\rho}_B(\mathbf{t}, \tau) = \frac{\#\text{ of occurrences of } \mathbf{t} \circ \tau \text{ in } \mathcal{C}}{\#\text{ of occurrences of } \mathbf{t} \text{ in } \mathcal{C}}$$

where $\circ$ is concatenation. Why *something like*? The question would be whether *exactly* this empirical value would be *generative* without an underlying structural model. Simply, suppose we have some $\mathbf{t}$, we generate a next $\tau$, but such that $(t_2, \ldots, t_B, \tau)$ is not in $\mathcal{C}$. Strictly speaking, we can't then use literal $B$-prefix conditional empirical distributions as they would not tell us what comes next. However we'll stick to the naivest possible calculation, and outline a well-defined sampling process below.

Clearly $\rho_{B-1}(\mathbf{t}, \tau)$ is a *marginal* relative to $\rho_B(\mathbf{t}, \tau)$, because $\rho_B$ is more specific than $\rho_{B-1}$. That is,

$$\rho_{B-1}(\mathbf{t}, \tau) = \mathbb{P}(\ t_B = \tau \mid t_1, \ldots, t_{B-1}\ ) = \sum_t \rho_0(t)\rho_B\big(t \circ \mathbf{t}, \tau\big)$$

where $\rho_0(t) = \mathbb{P}(T = t)$. Also

$$\hat{\rho}_0(\tau) = \frac{\#\text{ of occurrences of } \tau \text{ in } \mathcal{C}}{|\mathcal{C}|}$$

which we are assured is defined and positive (because the corpus covers the token language).

So how would we compute $\hat{\rho}_B$? To sketch a start, we can do this in a single pass over $\mathcal{C}$ as follows:

(0) Initialize a hashmap $\mathcal{P}_B$ whose keys are $B$-token strings and values are also hashmaps with single-token keys and whose values are floats (technically `doubles`). We will store $\hat{\rho}_B(\mathbf{t}, \tau) = \mathcal{P}_B[\mathbf{t}][\tau]$. Initialize $c = B$.
(1) Set $\mathbf{t} = \mathcal{C}[c - B : c)$, $\tau = \mathcal{C}[c]$
(2) If $\mathcal{P}_B[\mathbf{t}]$ does not exist, initialize $\mathcal{P}_B[\mathbf{t}]$ as needed, and set $\mathcal{P}_B[\mathbf{t}][\tau] = 1$. Otherwise, if $\mathcal{P}_B[\mathbf{t}][\tau]$ does not exist, set $\mathcal{P}_B[\mathbf{t}][\tau] = 1$. Otherwise increment $\mathcal{P}_B[\mathbf{t}][\tau]$.
(3) Increment $c$ and go back to (1) unless $c = |\mathcal{C}|$, in which case continue to (4).
(4) For all keys $\mathbf{t}$ defined in $\mathcal{P}_B$, compute

$$S(\mathbf{t}) = \sum_{\tau \in \mathcal{P}_B[\mathbf{t}]} \mathcal{P}_B[\mathbf{t}][\tau]$$

and update

$$\mathcal{P}_B[\mathbf{t}][\tau] \leftarrow \mathcal{P}_B[\mathbf{t}][\tau]/S(\mathbf{t})$$

Technically this is a bit like a double-pass algorithm considering the normalization in (4), but it is still $O(|\mathcal{C}|)$. We also might want to reverse the ordering of tokens in the keys of $\mathcal{P}_B$, to enable easier search with some tools that can do bulk return with partial key matching, but that's a detail. This is also embarassingly parallelizable, distributing the right overlapping subsets of $\mathcal{C}$ and merging globally, though we don't outline the details.

Now, we also need to reduce to $P_{B-1}, P_{B-2}, \ldots, P_0$ for any hope of prediction. Specifically, we could predict a token $\tau$ for which the concatenated subsequence $(\mathbf{t}[2:]) \circ \tau$ does not occur in the corpus. By assumption $P_0$ exists and is "complete", as the empirical frequencies of tokens in the corpus are defined by virtue of covering. That is, we can always simply sample from the simple occurrence likelihood. Our process could be to take find longest suffix of $\mathbf{t}$ that exists in the corpus,

$$\mathbf{t}[:-k] \quad \text{where} \quad k = \arg\min_{0 \le k \le |\mathbf{t}|} \{|\mathbf{t}[:-k]| : \mathbf{t}[:-k] \subset \mathcal{C}\}$$

and sample from $P_{|\mathbf{t}[:-k]|}[\mathbf{t}[:-k]]$. In the "worst case" $k = |\mathbf{t}|$ and we choose from $P_0$.

Prediction actually means a $\mathcal{T}$-set of suffix trees may be more suitable, where we access $P_B[\mathbf{t}]$ via following the "reverse" or suffix path

$$t_{|\mathbf{t}|} \to t_{|\mathbf{t}|-1} \circ t_{|\mathbf{t}|} \to \cdots$$

from a (guaranteed-to-exist) root $t_{|\mathbf{t}|}$ to the deepest accessible node. This could also aid in the "marginalization" process. Each node would contain a hashmap representing the distribution over next most likely tokens.

## 3. A Simple Example

## 4. Scaling

Mountain View CA
*Email address*: morrowwr@gmail.com