A Approximate Behavior of Metrics on Sequential Data

How do different metrics behave when used to measure autoregressive model outputs? Precisely answering this question is tricky and possibly analytically unsolvable, so we provide an approximate answer here.

Notationally, we consider N test data of length L (here, length is measured in tokens) with targets denoted $t_n \stackrel{\text{def}}{=} (t_{n1}, t_{n2}, ...t_{nL})$, the autoregressive model has a true-but-unknown per-token error probability of $\epsilon \in [0, 1]$ and the model outputs prediction $\hat{t}_n \stackrel{\text{def}}{=} (\hat{t}_{n1}, \hat{t}_{n2}, ...\hat{t}_{nL})$. This assumes that the model's per-token error probability is constant, which is empirically false, but modeling the complex dependencies of errors is beyond our scope.

A.1 Per-Token Error Probability is Resolution-Limited

Note that because we have N test data, each of length L, our resolution for viewing the per-token error probability ϵ is limited by $1/NL$. Here, resolution refers to "the smallest interval measurable by a scientific instrument; the resolving power." To explain what resolution means via an example, suppose one wants to measure a coin's probability of yielding heads. After a single coin flip, only two outcomes are possible (H, T), so the resolution-limited probability of heads is either 0 or 1. After two coin flips, four outcomes are possible (HH, HT, TH, TT), so the resolution-limited probability of heads is now one of $0, 0.5, 1$. After F coin flips, we can only resolve the coin's probability of yielding heads up to $1/F$. Consequently, we introduce a resolution-limited notation:

$$\lfloor a \rceil_b \stackrel{\text{def}}{=} a \text{ rounded to the nearest integer multiple of } 1/b \tag{3}$$

A.2 Token Edit Distance

We first consider an adaptation of the Levenshtein (string edit) distance for models that function on tokens rather than characters, an adaptation we term the *token edit distance*. The token edit distance between two token sequences $t_n, \hat{t_n}$ is defined as the integer number of additions, deletions or substitutions necessary to transform t_n into $\hat{t}_n$ (or vice versa).

$$\begin{aligned}
\text{Token Edit Distance}(t_n, \hat{t}_n) &\stackrel{\text{def}}{=} \text{Num Substitutions} + \text{Num. Additions} + \text{Num. Deletions} &(4)\\
&= \sum_{\ell=1}^{L} \mathbb{I}[t_{n\ell} \neq \hat{t}_{n\ell}] + \text{Num. Additions} + \text{Num. Deletions} &(5)\\
&\geq \sum_{\ell=1}^{L} \mathbb{I}[t_{n\ell} \neq \hat{t}_{n\ell}] &(6)
\end{aligned}$$

The expected token edit distance is therefore:

$$\begin{aligned}
\mathbb{E}[\text{Token Edit Distance}(t_n, \hat{t}_n)] &\geq \mathbb{E}[\sum_{\ell=1}^{L} \mathbb{I}[t_{n\ell} \neq \hat{t}_{n\ell}]] &(7)\\
&= \sum_{\ell=1}^{L} p(t_{n\ell} \neq \hat{t}_{n\ell}) &(8)\\
&\approx L(1-\epsilon) &(9)
\end{aligned}$$

The resolution-limited expected token edit distance is therefore:

$$\lfloor \mathbb{E}[\text{Token Edit Distance}(t_n, \hat{t}_n)] \rceil_{NL} \geq L\Big(1 - \lfloor \epsilon \rceil_{NL}\Big) \tag{10}$$

From this, we see that the expected token edit distance scales approximately linearly with the resolution-limited per-token probability. The real rate is slightly higher than linear because additions and deletions contribute an additional non-negative cost, but modeling this requires a model of how likely the model is to overproduce or underproduce tokens, which is something we do not currently possess.

A.3 Accuracy

$$
\begin{aligned}
\text{Accuracy}(t_n, \hat{t}_n) &\stackrel{\text{def}}{=} \mathbb{I}[\text{No additions}]\, \mathbb{I}[\text{No deletions}] \prod_{l=1}^{L} \mathbb{I}[t_{nl} = \hat{t}_{nl}] && (11)\\
&\approx \prod_{l=1}^{L} \mathbb{I}[t_{nl} = \hat{t}_{nl}] && (12)
\end{aligned}
$$

As with the Token Edit Distance (App. A.3), we ignore how likely the language model is to overproduce or underproduce tokens because we do not have a good model of this process. Continuing along,

$$
\begin{aligned}
\mathbb{E}[\log \text{Accuracy}] &= \sum_{l} \mathbb{E}[\log \mathbb{I}[t_{nl} = \hat{t}_{nl}]] && (13)\\
&\leq \sum_{l} \log \mathbb{E}[\mathbb{I}[t_{nl} = \hat{t}_{nl}]] && (14)\\
&\approx L \log(1 - \epsilon) && (15)
\end{aligned}
$$

Taking an approximation that would make most mathematicians cry:

$$
\begin{aligned}
\mathbb{E}[\text{Accuracy}] &\approx \exp(\mathbb{E}[\log \text{Accuracy}]) && (16)\\
&= (1 - \epsilon)^{L} && (17)\\
& && (18)
\end{aligned}
$$

This reveals that accuracy **approximately** falls geometrically with target token length. The resolution-limited expected accuracy is therefore:

$$
\lfloor \mathbb{E}[\text{Accuracy}] \rceil_{NL} = \lfloor (1 - \epsilon)^{L} \rceil_{NL} \qquad (19)
$$

From this we can see that choosing a nonlinear metric like Accuracy is affected significantly more by limited resolution because Accuracy forces one to distinguish quantities that decay rapidly.

A.4 ROUGE-L-Sum

Another BIG-Bench metric [28] is ROUGE-L-Sum [23], a metric based on the longest common subsequence (LCS) between two sequences. Section 3.2 of [23] gives the exact definition, but the key property is that ROUGE-L-Sum measures the “union” LCS, which means “stitching” together LCSs across the candidate and multiple references. As explained in the original paper: if the candidate sequence is $c = w_1 w_2 w_3 w_4 w_5$, and if there are two reference sequences $r_1 = w_1 w_2 w_6 w_7 w_8$ and $r_2 = w_1 w_3 w_8 w_9 w_5$, then $LCS(r_1, c) = w_1 w_2$ and $LCS(r_2, c) = w_1 w_3 w_5$, then the *union* -LCS of c, r_1, r_2 is $w_1 w_2 w_3 w_5$, with length 4. Intuitively, this disproportionately benefits models with smaller error rates because their mistakes can be “stitched” across multiple references; this is confirmed in simulation (Fig. 9).

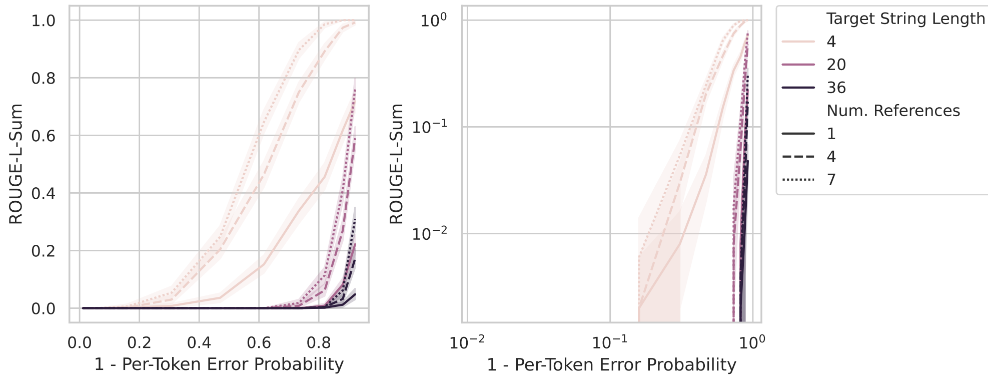

Figure 9: **ROUGE-L-Sum is a sharp metric.** Simulations show that as the per-token error probability slightly increase (e.g. from 0.05 to 0.1), the ROUGE-L-Sum metric sharply falls.

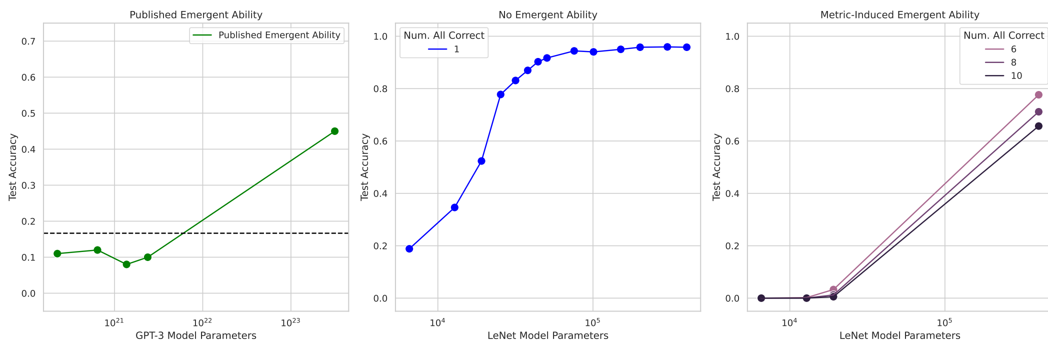

Figure 10: **Induced emergent MNIST classification ability in convolutional networks.** (A) A published emergent ability from the BIG-Bench Grounded Mappings task [33]. (B) LeNet trained on MNIST [21] displays a predictable, commonplace sigmoidal increase in test accuracy as model parameters increase. (C) When accuracy is redefined as correctly classifying K out of K independent test data, this newly defined metric induces a seemingly unpredictable change.

B Inducing Emergent Abilities in Networks on Vision Tasks

B.1 Emergent Classification of MNIST Handwritten Digits by Convolutional Networks

We begin by inducing an emergent classification ability in a LeNet convolutional neural network family [22], trained on the MNIST handwritten digits dataset [21]. This family displays smoothly increasing test accuracy as the number of parameters increase (Fig. 10B). To emulate the accuracy metric used by emergence papers [8, 33, 28], we use *subset accuracy*: 1 if the network classifies K out of K (independent) test data correctly, 0 otherwise. Under this definition of accuracy, the model family displays an "emergent" ability to correctly classify sets of MNIST digits as K increases from 1 to 5, especially when combined with sparse sampling of model sizes (Fig. 10C). This convolutional family's emergent classification ability qualitatively matches published emergent abilities, e.g., at the BIG-Bench Grounded Mappings task [33] (Fig. 10A).