A Caplan et al.'s (2020) Phonotactic Monkey

Caplan et al.'s Phonotactic Monkey is not free of functional pressures, since its assignment of word meanings is dependent on phonotactic effects that may already include functional pressures. Their phonotactic monkey is constructed as follows: an n-gram phonotactic model is fitted over word types; they sample (with replacement) from that model until a lexicon with N unique wordforms is generated. The token frequency of each word is set as the number of times each wordform was sampled from this phonotactic model. Each word type is then assigned a meaning. There are more meanings than words, and so, for unmatched meanings, word types are assigned to those meanings in proportion to their token frequency. This leads to the dependency structure in Fig. 4. Caplan et al.'s conclusion is that the phonotactic monkey generates evidence of efficiency, represented by negative correlation between word length and polysemy. However, as is depicted in Fig. 4, an *a priori* relationship is introduced between phonotactic probability (and indirectly length) and polysemy. Insofar as the relationship between these properties is itself functionally motivated, we argue that Caplan et al.'s model already has optimisation baked into it and is thus not free of functional pressures.

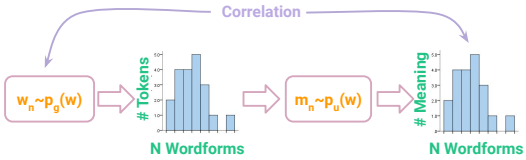


Figure 4: A graphical representation of the dependencies in Caplan et al.'s (2020) phonotactic monkey. A phonotactic distribution is used to sample tokens. The resulting token distribution is used to sample meanings. The meaning distribution, thus, is *a priori* correlated with the original phonotactic distribution. At the limit of infinite samples they will be identical.

B Monkeys' Algorithms

We present the algorithms used to sample text using our FCFS and PolyFCFS monkeys in Fig. 5.

```
def fcfs_monkey(corpora):
    for w_n in corpora:
        p_n = C_nat^{-1}(w_n) # sample |p(p)|
        if p_n not in C_fcfs:
            w' ~ p_g(w) # without repetition
            C_fcfs(p_n) = w'
        write(C_fcfs(p_n))
```

```
def poly_fcfs_monkey(corpora, δ):
    for w_n in corpora:
        m_n = bert([prefix_n ∘ w_n ∘ suffix_n])_n
        dists = (m_n − m_<n)^2
        if dists.min() > δ:
            w' ~ p_g(w) # without repetition
            p_n = n
            C_fcfs(p_n) = w'
        else:
            i = dists.argmin()
            p_n = P_fcfs(m_i)
        P_fcfs(m_n) = p_n
        write(C_fcfs(p_n))
```

Figure 5: Proposed random typing models' algorithms.

C Graphotactic Model

One of the key components of our monkeys is given by the distribution of $p_g(\mathbf{w})$, from which we sample wordforms. This graphotactics distribution is supposed to model the quality of a wordform for a specific language. Consider the contrast between *brick*, *blick* and *bnick*. Only the first of these is an actual wordform of English, whereas the first two are graphotactically good and judged plausible by English speakers. *Bnick* is an example of a graphotactically implausible word; if an English speaker encountered it in some text, they might believe it to be misspelled.

How to best model a graphotactics, or phonotactics, distribution is debated (Hayes and Wilson, 2008; Goldsmith and Riggle, 2012; Futrell et al., 2017; Pimentel et al., 2020b; Mayer and Nelson, 2020). While much work in the computational linguistics realm relies on n-grams for this (e.g. Dautriche et al., 2017; Mahowald et al., 2018; Caplan et al., 2020), we make use of LSTM character-level models, as they are expected to produce more reliable distributions; see Pimentel et al. (2021a) for a longer exposition on why better graphotactic models are important for such an analysis.

Formally, we first encode each character $w^{(t)}$ in a lookup embedding $\mathbf{e} \in \mathbb{R}^{d_1}$. We then feed these character-level embeddings into an LSTM module (Hochreiter and Schmidhuber, 1997), which gives us the high-dimensional contextual representations $\mathbf{h} \in \mathbb{R}^{d_2}$. Finally, we linearly transform these representations and feed them into a softmax to obtain a distribution over next characters, modelling:

$$p_g\left(w^{(t)} \mid \mathbf{w}^{(<t)}\right) = \text{softmax}(\mathbf{W}\,\mathbf{h} + \mathbf{b}) \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{|\Sigma| \times d_2}$ and $\mathbf{b} \in \mathbb{R}^{|\Sigma|}$ are trainable parameters. We train this architecture on a set of word types for each language, minimising this model's cross-entropy using gradient descent. Our model has 2 hidden layers, an embedding size of 128, a hidden size of 256 and we use a dropout of .33. Further, we optimise our model with AdamW (Loshchilov and Hutter, 2019) as implemented in PyTorch (Paszke et al., 2019).

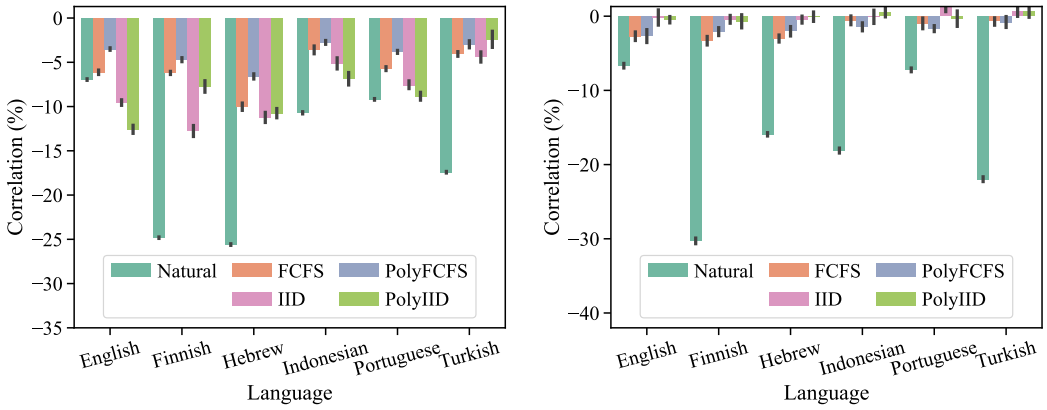
Sampling plausible wordforms from such a model is also not a straightforward task. While we could sample these wordforms directly from the distribution learned, this could lead to unrepresentative forms. The use of a softmax non-linearity, for instance, prohibits this model from placing a zero probability on any specific character transition. This, combined with the multiplicative composition of probabilities, could lead to implausible, but short wordforms, such as *zyz*, being assigned higher probabilities than long, but plausible ones, such as the real Portuguese word *inconstitucionalissimamente*.[7] In order to reduce these issues, we first introduce a minimum threshold of 0.01 probability for any transition $p_g(w^{(t)} \mid \mathbf{w}^{(<t)})$, setting its value to zero if it falls below that. Secondly, we use a temperature of 0.5 when sampling words, causing our distributions to become more peaked for the more likely transitions:

$$p_g\left(w^{(t)} \mid \mathbf{w}^{(<t)}\right) \propto \begin{cases} 0 & \text{if} < 0.01 \\ \text{softmax}\left(\frac{\mathbf{W}\,\mathbf{h}+\mathbf{b}}{0.5}\right) & \text{else} \end{cases} \tag{2}$$

When sampling wordforms for our i.i.d. monkeys, however, we get unrealistic homophony rates when using the above parameterisation. Specifically, while only roughly 4% of wordforms are estimated to be homophonous in natural languages (Dautriche, 2015), we get an average of 33% homophonous wordforms when using the above sampling scheme with repetition. Further, Trott and Bergen's (2020) results suggest that each wordform has, on average, roughly 0.1 homophones;

[7] This adverb means something done `in a completely unconstitutional manner`.

however, the above sampling scheme leads to an average of 2.4 homophones per wordform. We thus change the temperature we use on our i.i.d. monkey baselines to 5, leading to an average of roughly 0.09 homophones per wordform; this is much closer to what Trott and Bergen (2020) results suggest. We nonetheless also present the results when sampling i.i.d. with a temperature of 0.5 in Fig. 6.



(a) Frequency–length correlations.

(b) Polysemy–length correlations.

Figure 6: Double Column Figure: Natural and simulated lexicons' correlations when using a temperature of 0.5 in IID and PolyIID random typing models. In this case, IID and PolyIID simulated lexicons have on average 2.4 homophones per wordform (natural languages have roughly 0.1) and a single wordform in these lexicons has up to 544 homophones (natural languages have 20 at most, on average). Natural frequency–length correlations are still significantly the strongest in 4 of the 6 languages, and Natural polysemy–length correlations are significantly stronger in all languages.

D Pimentel et al.'s (2020a) Polysemy Metric

Pimentel et al. (2020a) proposed a new information-theoretic interpretation of lexical ambiguity, where lexical ambiguity can be quantified as the conditional entropy of meanings given a specific word type, i.e. $\mathrm{H}(S \mid W = \mathbf{w})$. Under this definition, then, lexical ambiguity is defined as the uncertainty left about word meanings once we are given a word type in isolation. This is a general framework, which can be applied on different ambiguity estimates. For instance, given the number of synsets in WordNet, and assuming a uniform distribution over them, we can write:

$$\mathrm{H}(S \mid W = \mathbf{w}) \approx \log_2(\#synsets(\mathbf{w})) \tag{3}$$

More importantly, however, is that the formalisation in Pimentel et al. (2020a) allows us to estimate the polysemy-level per word type not only on naturally occurring text, but also on random typed text. To compute the lexical ambiguity of a word type using BERT, we estimate:

$$\mathrm{H}(S \mid W = \mathbf{w}) \approx \log_2 \det(2\,\pi\, e\, \Sigma_{\mathbf{w}}) \tag{4}$$

where $\Sigma_{\mathbf{w}}$ is the covariance matrix taken on word type $\mathbf{w}$'s BERT embeddings.[8] This estimate

[8] In our implementation, we instead compute this determinant over the diagonals of the co-variance matrix $\Sigma_{\mathbf{w}}$. This improves correlations with the number of synsets in WordNet, and is equivalent to assuming BERT's neurons are independent.

assumes a Gaussian distribution over a word's contextual embeddings. In the likely case where these values are not actually Gaussian-distributed, this becomes an upper bound on the true entropy.

Sanity-check Results. To validate this measure, we derived polysemy estimates using both WordNet (Miller, 1995; Noor et al., 2011; de Paiva and Rademaker, 2012; Lindén and Carlson., 2010; Bond and Paik, 2012) and this BERT-based metric (per word type) in 4 of the natural languages in our sample, and then computed the correlation between these metrics.[9] These values are presented in Tab. 1. All analysed languages present a significant, albeit modest, positive correlation between both polysemy estimates. All languages also show the predicted negative relationship between the BERT-based measure of polysemy and length. When polysemy is estimated with WordNet, both English, Finnish, and Portuguese still show the expected relationship, but the opposite trend is in place for Indonesian.

	Natural			
Language	# Types	WN vs. L	B vs. L	WN vs. B
English	6771	-0.28‡	-0.07‡	0.46‡
Finnish	2045	-0.13‡	-0.30‡	0.22‡
Indonesian	4180	0.15‡	-0.18‡	0.31‡
Portuguese	3233	-0.09‡	-0.07‡	0.29‡

Table 1: Validation of our polysemy estimates. Values in last three columns are Spearman correlations between: WN, per-word polysemy estimates using the log of the number of synsets a word type has in WordNet; B, per-word BERT-based polysemy estimates; and L, words' lengths. ‡ means statistical significance at a level of $\alpha = 0.01$.

[9] We were unable to run the experiment for Turkish, for which WordNet is not available, and Hebrew, where we faced technical issues due to a mismatch in the diacritics used by Wikipedia and WordNet.